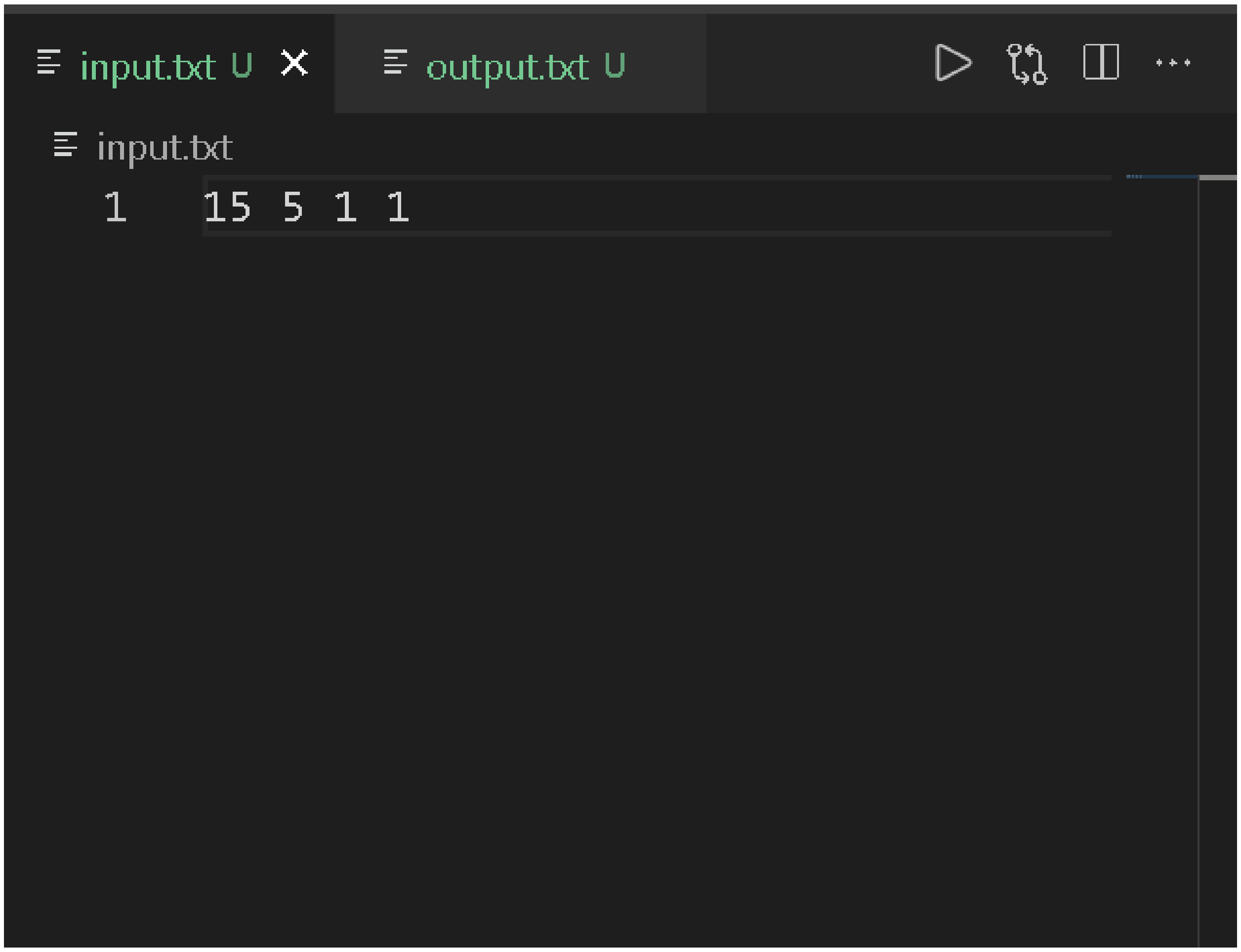
DAA Assignment

By Ayush Varma
1910110108

Input File



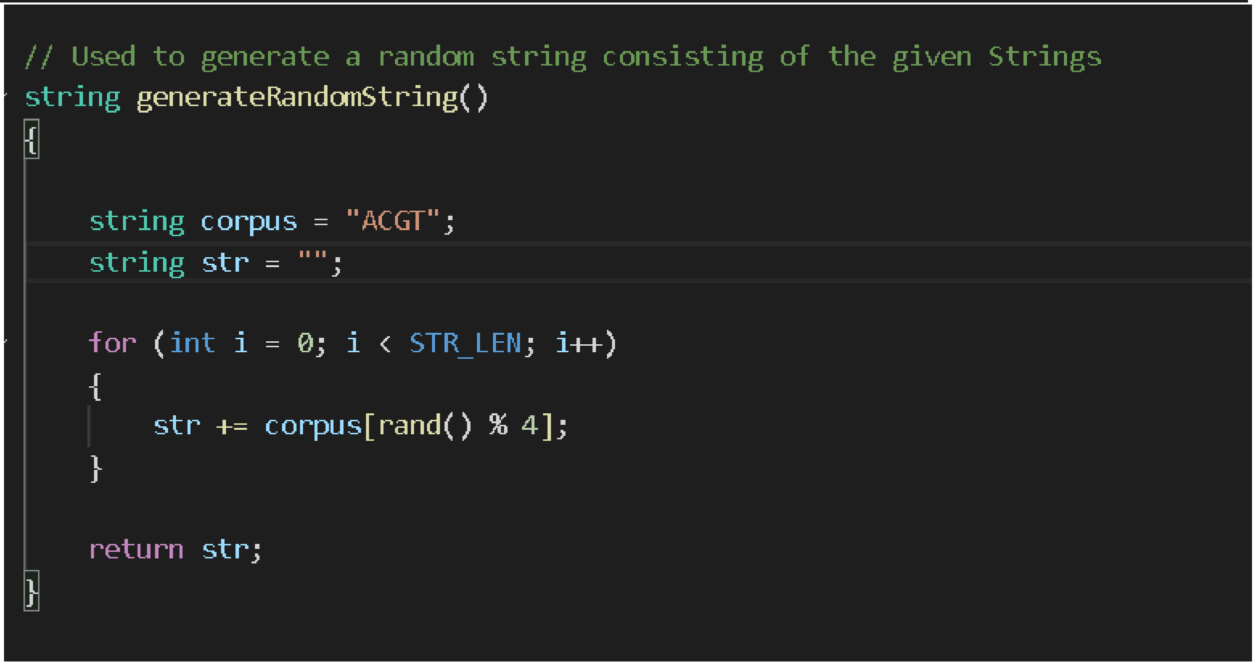
```
15 5 1 1
```

Output File

Algorithm Analysis

The program consists of several functions used for different calculations.
Here are the different types of functions used along with their usage:

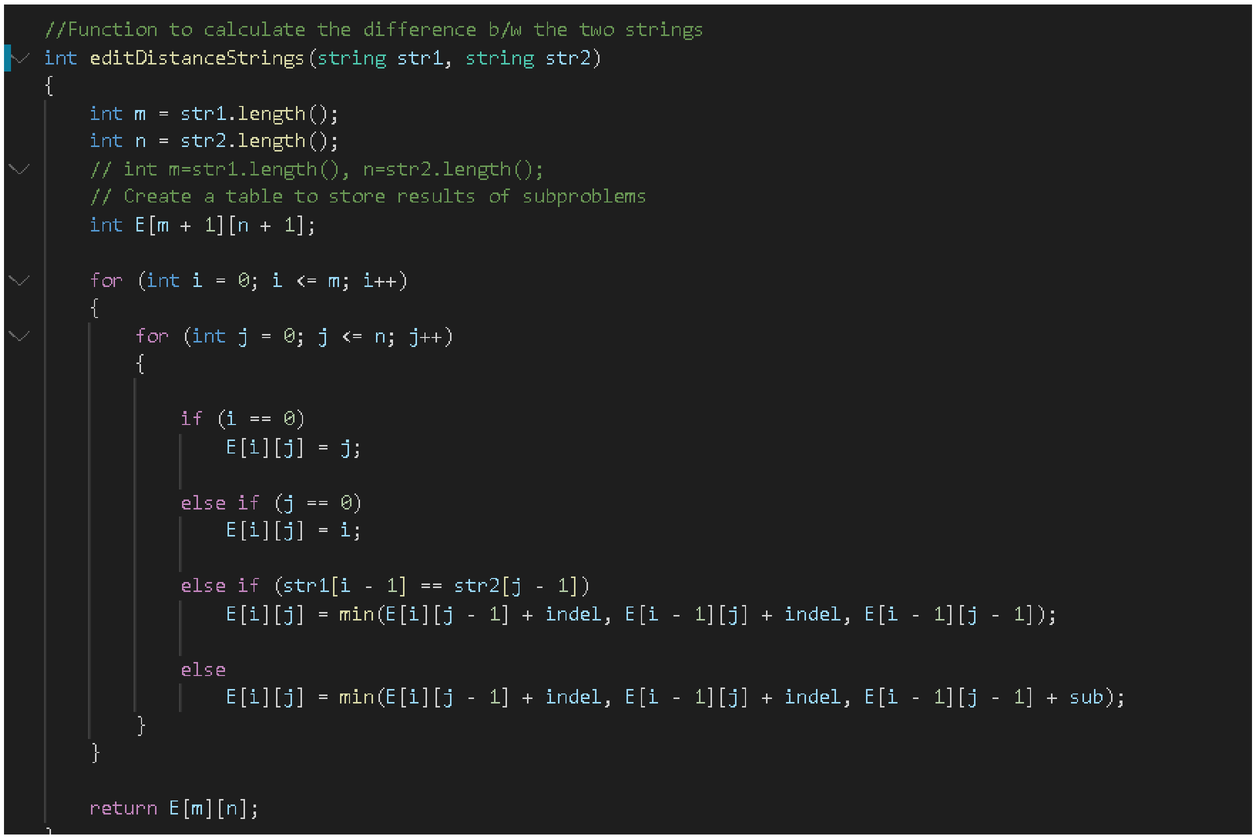
```
1)generateRandomString()
```



```
// Used to generate a random string consisting of the given Strings
string generateRandomString()
{

    string corpus = "ACGT";
    string str = "";

    for (int i = 0; i < STR_LEN; i++)
    {
        str += corpus[rand() % 4];
    }

    return str;
}
```

This function is used to create a random string from the corpus of 4 characters, we use random function to do so by using rand() function along with the srand() function by using time as a seed to create a unique string everytime, since rand generates a redundant output.

```
2)editDistanceStrings()
```



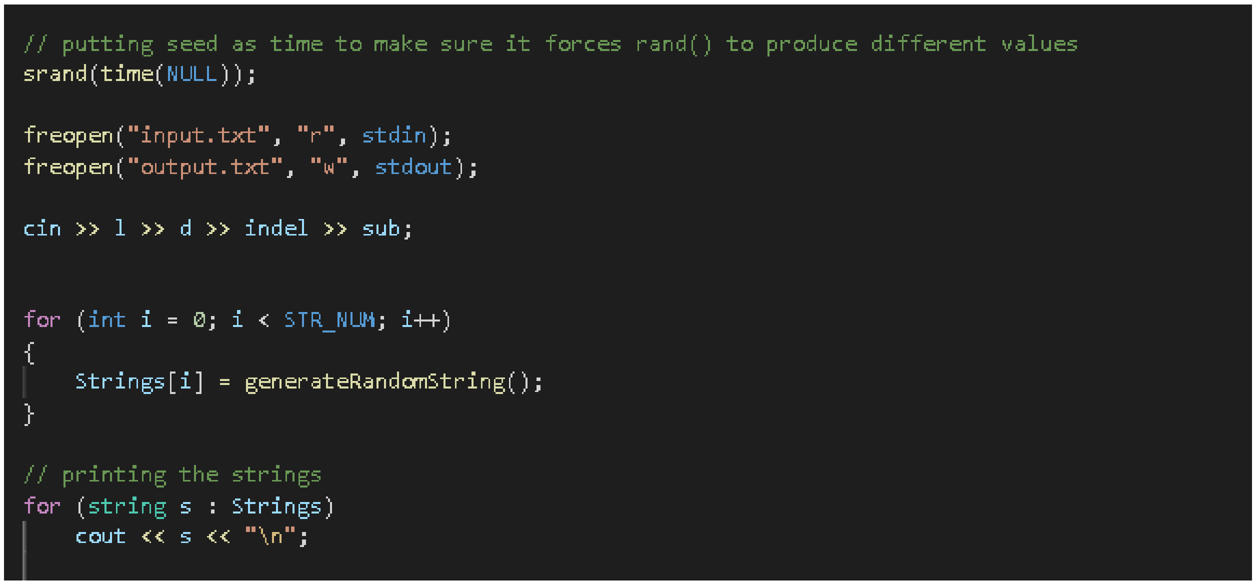
```cpp
//Function to calculate the difference b/w the two strings
int editDistanceStrings(string str1, string str2)
{
    int m = str1.length();
    int n = str2.length();
    // int m=str1.length(), n=str2.length();
    // Create a table to store results of subproblems
    int E[m + 1][n + 1];

    for (int i = 0; i <= m; i++)
    {
        for (int j = 0; j <= n; j++)
        {

            if (i == 0)
                E[i][j] = j;

            else if (j == 0)
                E[i][j] = i;

            else if (str1[i - 1] == str2[j - 1])
                E[i][j] = min(E[i][j - 1] + indel, E[i - 1][j] + indel, E[i - 1][j - 1]);

            else
                E[i][j] = min(E[i][j - 1] + indel, E[i - 1][j] + indel, E[i - 1][j - 1] + sub);
        }
    }

    return E[m][n];
```

The above function is used to calculate the difference between the two strings, it calculates the difference between the two using the difference of characters at each place as well as the steps required to change the former string to the latter.
I have used the Dynamic Programming approach to solve the distance b/w the two strings since on observing the algorithm, It was observed that it contained subproblems that satisfied the fundamental properties of dynamic programming i.e
1) Overlapping Subproblems
2) Optimal Substructure

Time Complexity: O(m x n)

Auxiliary Space: O(m x n)

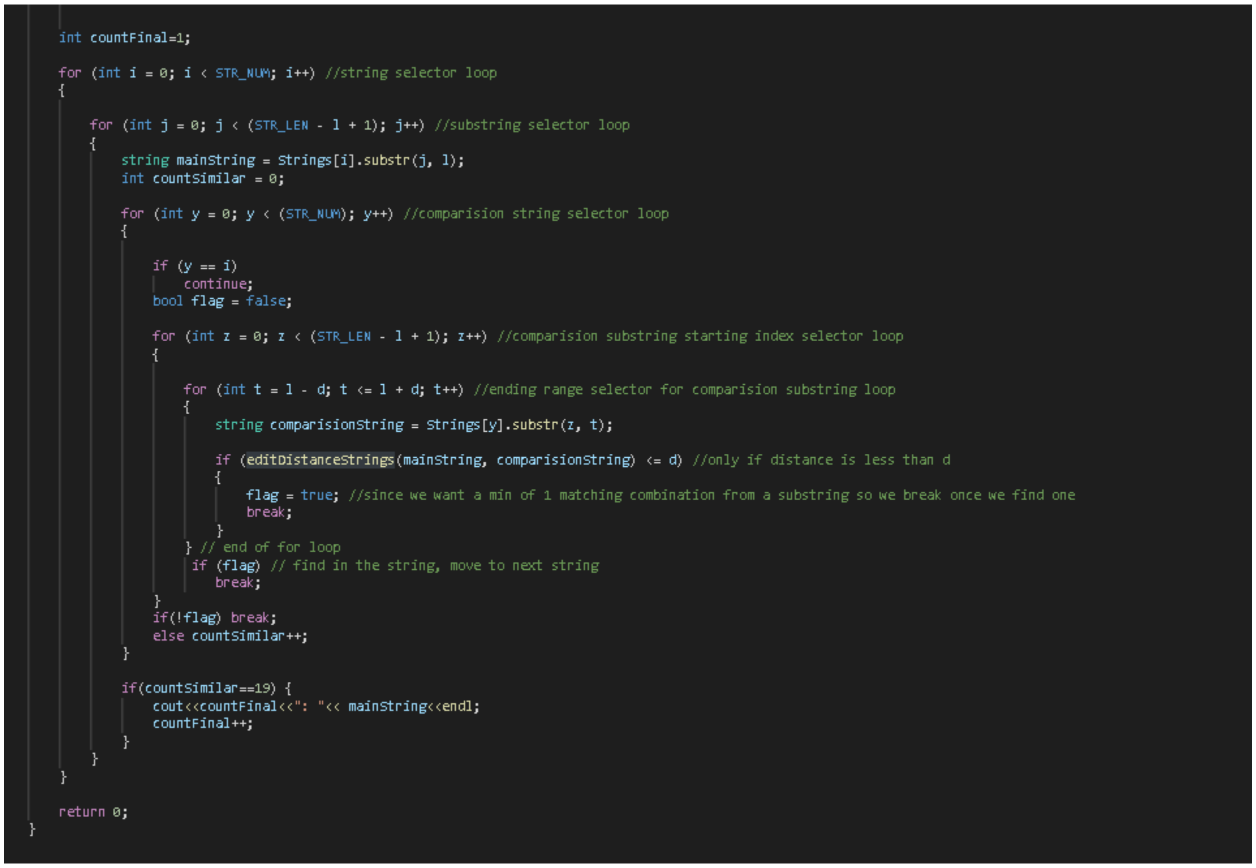
3)Main Driver Function



```
// putting seed as time to make sure it forces rand() to produce different values
srand(time(NULL));

freopen("input.txt", "r", stdin);
freopen("output.txt", "w", stdout);

cin >> l >> d >> indel >> sub;


for (int i = 0; i < STR_NUM; i++)
{
    Strings[i] = generateRandomString();
}

// printing the strings
for (string s : Strings)
    cout << s << "\n";
```

The above snippet is used to read the input file which contains the input of l, d, indel and sub and later it calls the random generate string.

4) Main Driver Function 2()



```
    int countFinal=1;

    for (int i = 0; i < STR_NUM; i++) //string selector loop
    {

        for (int j = 0; j < (STR_LEN - l + 1); j++) //substring selector loop
        {
            string mainString = Strings[i].substr(j, l);
            int countSimilar = 0;

            for (int y = 0; y < (STR_NUM); y++) //comparision string selector loop
            {

                if (y == i)
                    continue;
                bool flag = false;

                for (int z = 0; z < (STR_LEN - l + 1); z++) //comparision substring starting index selector loop
                {

                    for (int t = l - d; t <= l + d; t++) //ending range selector for comparision substring loop
                    {
                        string comparisionString = Strings[y].substr(z, t);

                        if (editDistanceStrings(mainString, comparisionString) <= d) //only if distance is less than d
                        {
                            flag = true; //since we want a min of 1 matching combination from a substring so we break once we find one
                            break;
                        }
                    } // end of for loop
                     if (flag) // find in the string, move to next string
                        break;
                }
                if(!flag) break;
                else countSimilar++;
            }

            if(countSimilar==19) {
                cout<<countFinal<<": "<< mainString<<endl;
                countFinal++;
            }
        }
    }

    return 0;
}
```

This is main driver function which handles all the complexities of the program as required by the questions. This function first starts with selecting the string to be checked for comparison and the next loop starts to select a substring from the given string.
The third loop is used to select the comparison string, while the fourth and the fifth loop yields us the starting and ending index loop boundaries to be used for the checking of the strings.

The complexity of the program is of the order O(n^5) since it uses five nested loops to solve the DNA sequencing problem.

Improvement

Since the time complexity of the program is of very high order therefore it can be further reduced and hence made faster. Using better traversal techniques can also lead to faster and better computational time since it would give us a better approach to finding answers to such lengthy sequences of high orders. Also, the use of computers with high computing power would yield a better and faster result.