Algoritmo LCS

Este reporte no es de vital importancia para la práctica, sin embargo, decidí crear uno ya que es necesario explicar algunos cambios que se hicieron:

1.- Método llenarVector

El código original es el siguiente:

```
void llenarVector(char M[], String cadena) {
    int i;
    char a;
    a = cadena.charAt(0);//getchar();
    M[1] = a;// A partir de aqui se empezara a llenar Los Vectores
    i = 2;
    while (i < cadena.length()) {
        a = cadena.charAt(i - 1);
        M[i] = a;
        i++;
    }
    //M[i-1]='\0';
}
```

El código modificado por mí es el siguiente:

```
void llenarVector(char M[], String cadena) {
    int i;
    char a;
    i =1 ;
    while (i <= cadena.length()) {
        a = cadena.charAt(i-1);
        M[i] = a;
        i++;
    }
}
```

El cambio se debe a que no se tomaba el ultimo carácter de la palabra ingresada, ademas de que ahora el ingreso del primer carácter se hace en el while, este método provocaba que no se tomara la palabra completa provocando que al momento de probar con palabras que se usan como ejemplos no dieran el mismo resultado.

2.- Método ejecutarAlgoritmoLCS

El código original es el siguiente:

```
void ejecutarAlgoritmoLCS(char X[], char Y[]) {
        int m, n, i, j;
        m = X.length - 1;
        n = Y.length - 1;
        for (i = 1; i <= m; i++) {
            c[i][0] = 0;
        }
        for (j = 0; j <= n; j++) {
            c[0][j] = 0;
        }
        for (i = 1; i <= m; i++) {
            for (j = 1; j <= n; j++) {
                if (X[i] == Y[j]) {
                    c[i][j] = c[i - 1][j - 1] + 1;
                    b[i][j] = '/';
                } else {
                    if (c[i - 1][j] >= c[i][j - 1]) {
                        c[i][j] = c[i - 1][j];
                        b[i][j] = '|';
                    } else {
                        c[i][j] = c[i][j - 1];
                        b[i][j] = '-';
                    }
```

```
            }
        }
    }
}
```

El código modificado por mí es el siguiente:

```
void ejecutarAlgoritmoLCS(char X[], char Y[]) {
    int m, n, i, j;
    m = X.length-1;
    n = Y.length-1;
    c=new int[m+1][n+1];
    b=new char[m+1][n+1];
    for (i = 0; i <= m; i++) {
        c[i][0] = 0;
    }
    for (j = 0; j <= n; j++) {
        c[0][j] = 0;
    }
    for (i = 1; i <= m; i++) {
        for (j = 1; j <= n; j++) {
            if (X[i] == Y[j]) {
                c[i][j] = c[i - 1][j - 1] + 1;
                b[i][j] = '/';
            } else {
                if (c[i - 1][j] >= c[i][j - 1]) {
                    c[i][j] = c[i - 1][j];
                    b[i][j] = '|';
                } else {
```

```
                c[i][j] = c[i][j - 1];
                b[i][j] = '-';
            }
        }
    }
}
```

El único cambio es que los arreglos c y b se le asignan los valores que tendrán con base en la longitud de las palabras que se ingresan mediante estas líneas:

```
c=new int[m+1][n+1];
b=new char[m+1][n+1];
```

Hay que mencionar que, gracias a este cambio, se cumple con la tarea, que es llevar al máximo este algoritmo, para ello en lugar de dar un valor predefinido a MAX_X y MAX_Y ese valor será el valor de la longitud de la cadena B y cadena A respectivamente más uno. Así se puede explotar el algoritmo al máximo.

3.- Método imprimirLCS

El código original es el siguiente:

```
void imprimirLCS(char X[], int i, int j) {
    if (i == 0 || j == 0) {
        return;
    }
    if (b[i][j] == '/') {
        imprimirLCS(X, i - 1, j - 1);
        System.out.print(X[i]);
    } else if (b[i][j] == '|') {
        imprimirLCS(X, i - 1, j);
    } else { // "-"
        imprimirLCS(X, i, j - 1);
    }
}
```

El código modificador por mí es el siguiente:

```
void imprimirLCS(int tamA, int tamB, char [] A) {
    int i = tamA;
    int j = tamB;
    ArrayList<Character> ImprimirSubsecuencia = new ArrayList();
    while (i >= 0 && j >= 0) {
        if (b[i][j] == '/') {
            ImprimirSubsecuencia.add(A[i]);
            i--;
            j--;
        } else if (b[i][j] == '|') {
            i--;
        } else { // "-"
            j--;
        }
    }
    for (int k = ImprimirSubsecuencia.size() - 1; 0<=k; k--) {
        System.out.print(ImprimirSubsecuencia.get(k));
    }
}
```

Este cambio se realizó debido a que la versión original lo hacía de manera recursiva provocando que a mayor sea el tamaño de los strings de entrada más tardado seria su ejecución, sin embargo, de manera iterativa ya no es tardado realizar esa funcion, de hecho, es mucho más optima.

Estos fueron los cambios necesarios para su funcionamiento, importante mencionar que la TABLA DINAMICA se omite debido, de igual manera, al tamaño de los strings de entrada y que en lugar se usar un Scanner para la entrada de texto se decidió optar por el uso del BufferedReader, debido a que es el óptimo para cadenas de gran tamaño.

A continuación se deja un ejemplo usando como Cadena A a la primer secuencia genómica de este enlace:

https://ftp.ncbi.nlm.nih.gov/ReferenceSamples/giab/data/AshkenazimTrio/HG002_NA24385_son/PacBio_CCS_10kb/m54238_180628_014238.Q20.fastq y como Cadena B a la segunda secuencia genómica.

```
 ======== -- -- ------------ ----- === ----- --- ------------ ========
 | Problema de la SubSecuencia Comun Mas Larga Con Programacion Dinamica |
 ======== -- -- ------------ ----- === ----- --- ------------ ========

       ==> Ingrese la Cadena A: AAAATTGGAGATACCACAAAGGCTGAGGATAATTTATACAGTGAGATCATAGGTAAGCAAGATATAAAACCCATGCACAGATGTGAGACTTAACGATCAGGAAAAATCATTTCTTCTTTAAGACAGAGGTAA
       ==> Ingrese la Cadena B: AAAGGTGGCGGGGGAATGACGTGGGCACAGTGGCTCACACCTATAATCCCAGCACTTTGGGAGGCTGAGGTGGGCAGATCAAGACCATCCTGGCCAACATGGTGAAACCTCACCTCTACTAAAAATACAAAA

Se omite la tabla dinamica por razones de tamaño

 ==> La Subsecuencia comun mas larga B en A:

 ==> { AAATGGAATACCACAGGCTAATAAATACAGGAGCTAGGTGCAGATAAACCATCCTGGCAACATGGAAATCATCTCTAAAAATAAAAATAGCTATGTACATTTTATCATTTAGTATTCTTCACACAGTATAGTGGACAGATGCACTGTAGAGAATCTAA

 ==> El numero de ocurrencia es: 5897
```

Y la subsecuencia sigue y sigue, hasta no poder verla en la terminal de NetBeans, sin embargo, ocupando la consola de Windows obtenemos lo siguiente:

```
 ==> La Subsecuencia comun mas larga B en A:

 ==> { AAATGGAATACCACAGGCTAATAAATACAGGAGCTAGGTGCAGATAAACCATCCTGGCAACATGGAAATCATCTCTAAAAATAAAAATAGCTATGTACATTTTATCATTTAGTATTCTTCACACAGTATAGTGGACAGATGCACTGTAGAGAATCTAAAATAAAA
AAAAAAAAAGCTCCATCTCACGCCGTTTCAGACTTTGGAGCCGGCAGGGGGTCATCAGGGTTAGCGGCCCTAAAATTTACTAAAAATAAAAAATTAAGCATGAGGTGCCTTAATAGCTATCAGGGTAGAGGAAATTTTACCGGACGAGGTTGATAAATGTCTATCTAAGAAG
ACTCTTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATTGCTAGTGGTACGTGTAATCAATTTGGGAGAAGGCAGGCGATACTTAATCAAGTTCGCACCTGGAACAGCAAATTTATAAAAATACAAAAATTAGCCCAGATGCAGATGTAATCCCATATGAGATGAGAGGAGAATCC
TTAACCGAGCGGAGGTTGAGTGAGCCGAGATACACACTCATCAGCCTGGCACAGACAACTGCATCTAAAACAAAACAACCAAAAACCACATCATATAAAACTCTAATATTTGCTGAAAATCACTTAAAGTTCCAGGGCCGAAAAGGATTTATGTAAAACAAAATTAAGCAGT
TTTCTTATTAAAATACTCAAAACAACATAAAATCTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTGGTTATGGATTTTTTTTTTCTGGATGAAACAAATGGTATGTCCAGGTTCAAGTTTTGCTCATCTGAGTAGGGGATTATATCCAGAAGCTGGTAATTTTTGATTTTTAGT
ATATGGGTTTTTGCCATTTGGCCACTGTCTTGAACTCCTGACCCAGGGATCCCCTCCCCCCCCAAATGCTTGGTTCAGCATGAGCACACGAGCCAAACTGTTTAAACAAAACGAAAACAAAACAAAAAAAAAAAAAATAATCTTATTGGGGAATAAAGAACACCAAAATGTAA
AAATAAGTAAATATAAAAACTTTTTTTTAATTGGTCACGTATAATCTCAGTATTGGGAAGCTGAGGGGGGCGATCTGCTCTTCACCAGCTGGACAGGCGAACCGCTCACTAAATCAATCCGGCAGCAATCCCTGTACTCAGTGCTGGGAGGTGACAAAGTCATTTGAGGGAG
GCAAGTTGCAGTGGCTGAATTGCCCATGACTCCAGCCTAGGTGACAGGCAAGACCCACCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGTTTTTAATTTTTAATCTCTTTAAAGATAATACTTTTAAGGAAAAAATATTACTGTAGGTTTTTCATATTAGAAGTAAAAGTAGACAAA
AATAGACAAAATGGGAAAGAAATAAATTATTTTGTCTTAACTAATGTTAAGGATATTTTTAAAGGTAGGTTAAGAAGTTAAAATATATTTAAACCTGAAACTAATAAATTAAAATAAAAACAAAAGGATACTAATAGCTAACAGTGACGAAAAAGGAAATCATAAAAAAAAA
AATGGCGGCAGTGGCTCACACCTGTATCAGCACTTTGGAGGCAAGGGCGATCACCTGAGGTCAGAGTTGAGACAGCCTGGCCAACATGGTGAAACCGTCTCTACTAAAAACAAAAATTAACTGTGTGGTGCGGGCCTGATCAGCTACTCGAGGCTGAGGCAGGAGAATGCTT
GAACCCGGAGGGGAGGTTGCAGTGAGCCAGATGGCCATGCCCCACCTGGGCAAGAGCAACTCTCTCAAAACAAAAAAAAAAAAAATTTTTCTCATTTTTAATGAAGTTGTATATATATTCTCATTGCCAGCTAGTCTCAACTCCTGCCAGCCAAACACCCTTGTTTCTTTAGC
TTTGCACCAATATTTTGGAGAAACGGTGAATATATCGATTAGATTACCAAAAAACCTCCATGTAAATTTATATTTCCTGTGTGCAATTCCTTTTTCCAATTATGAGACAAACCCTCCATCAGTAGTTCTGAAGCAATTTATCTATTAAATAAAAAAGAACCTAGAAGTTATT
TGTCCCCTTCTTCCCCCGACTAGAAATGCCAGATGGTGTCCATTTTTCCGCACTTCCTACACCTGTCCCTAAACTCTTCTTGGGTTTTTTTCCAGGTATATGCTTCTGCAAAGATTACAGTTACGTTTGTGGGATGAACAAACTTATTTTTTCATTTTTTTTTTTTTGACAT
TTTCTTAGCTCAGTGAAAAAGGAGAGCTGGGTGGTCCAGCCCCAGAGCATTTCAGACCCAAGGGCAATTTAAACAGAAGGTTTCTAGTTGGAAGCTTTCCTCCTTCTGATCAAGGTAAAAGAACAATAGCACCACTCTGGCTTAGCAATAATAGGATTGCCAGTGAATATAC
TAAAAAGGATAGAGTCAGGGACAGGAGGTGGGGTCTGAAGAATAATAAAAGGGGGGCCGAATGCTACATTGTATCTAGCATTTGAGAGTGAGGGAGGAATCACTTACCAATTTGGACCCCAAAATAGTATTTCTTACTAAAAAAAAAAAAAAAAAATTTAATTAAAAAAATA
AAAAGTTTTAGTGAAAACCGTTTCAACACTTATTGTATACAAATGGTGAAAATGATCTTTCTTCATCTTAAAAGGATTCAATAAGTGCCTCTTTATGTTGGTTTAACCTCCTCAAGATTTTTGAAGGAATGGTGGTCCTCTGAATCAACAATTTTATAAAAGAAAAACAGGC
TGGCCATTGAAATGTAATTAACACTTTGGAGGCAGTGGGGATCACTGAGCCAGGAGTTGAGCCTGGCCAACATGGGAAACCCTCTCTACAAAAATAAAAAATTGCCAGGTGTTGTGCCATCATCCCAGCTATGGAGCTGAGGGGGATGTAACCCGGGTGGAGATTGAGTGAC
AGATTGCCACTGACTCCAGGCCAGATGCTCAAAAAAAAAAAAAAAAAAAAGTAAAAGAGATTTCACTGTTAAACAGGCCACCTCTTGGACACATATCAAAAACTAATTGCCCTCCGTTTTTTTACCTTGTTCTGCATAATATAATGCAAAAAAAACAAGATTGAAAACTAAGGAA
CTCAGCACAGTGACGACTATTTTTTAAAATAGCCAGCAGAACCATTACTGTTTTGAAGGAAGATGTAATGTTATGAAGGTATTTCACATGTGGAAGGGAAAAATCTGGAAGAAAAACATGAGAAAAGGTTGCAGATTTCAGGCTTTTTTTAGTTTTAACCAGGTTACAACTG
AGGGAAAAATTACATATGTGGTTCAGTTTGAAAATAATGCTAGCACAAAAATTTTTCAAATGTTTAACAAGTTTTTTTCCAAGAATTTTTTTACATAATATTTTAACTAATCACAATAAAGTCTTTACATTCAAGTAGTAACTTTTATATTTCTAAATGTTAATATTCTGTT
TTTACCAGACTATTCAGCAAAAACAACCAATTTTTACAGCTTTCTTTTATAGTGAACAAATATATTTCCAGATAAATTTGAAGCAAAGTGATGACAAATTTTCCTGTGGCTCAGCATGAGTGAAGATCATAGTGACTCAGAATCAAAACTGTTATTTCCAACTTTCAAAGAA
AAAATAGGAGATGAAAATTTGCATTATTGCAGTGGGCAGTCTAAAGTTTAATTTGTCACAAAATTTTAAAAATGTAAATAAATATTTCTCTTCAAGTTATTTAATTTTTTCCTCGTTGTTGTGTTTTTCAAGAAGAGGATTTTACCAGTAGCAGGGACAACCCAGTACTACC
CTCCCCTGTTAAGCAACATCACACGGTTTTTTTTATTTATAGAGATAGTTTACCCAGCAATCCTAATAAATGATTGTGGTCAAAATGATAAAACCACTTCAGAAGTAAATCATATTTTTGTGTTTGAATTTTGGGAGCGAGGCAGGAGATTTTGACAATAATCAGCCTGTGG
TAAACCCCTTTACTAATCAAAATTGCCTTAACCTGTATCAGTACAGGCTACAAGAATTTATTCACAGAGATTGCTCTAGATTGTCACTGATGCTCTGTTAACCACATCCTTGGCCTCATTCACGCCTACACATTTACAGGTATATATGGGGTGCCAGCCTAACATGTCTGTC
TCTCTAAAATACAACTGTGTTTCCGAACTCATACCAGAGCTCAAGGCTGAACCTAGGATTCATCCAGTGCCCTTTCCTGTCCATCCACTCAAGTTGTATCTTGAATCTTTTTGTCTGTTGTACACAACAAATGGAGAACATTTGTAAAGAAGTAAATAGCACAGAGGTGCCA
TTTGTTTTAATTTGATCTCATAAATCAGCACAGGCAGAAGAGCTCATAAAAATAAATAGATCAGTCTCTTGTAATGAGTTTGCTCTATTGCACTTAAGAATAAAGTTTTCCATGCTCTACACATAATCACACTTTAGGCCCAGCATCTGTCATTCCCAGCCTGCATCTTAAA
CCCCTTTTTTTTCTCTGGGGGCGCCTGTCCCCACTGGAGTGAGGGGGATCGCTTGAACTTATAGTTCAAGCGATTCCCTGCCTCAGCCTGGGCAACAGGCACCCCCACCAAACAGTTATTTTTTATTTTTAGGGTGGTGTTTCACCATTTGGCCGGTGGTATGAATCGATTG
AACCACCCAATAGCCTCCCAAATCATTACGGCGTGGCCACCATCCCAGCCGCTAAACTTTAACTGAGGTAGTTAGTGACAATTTCTGATCCCTAAATAGCTGTCCACACAGATCTTTAACATTTCACCAAAATTATATGTATTGTTCTGTTTTTTTTTTTTTTATTCTCTCT
TCCTAAATATATCAACTATATTAATTTCATTTTTTTATCAGAGCAAACCTCTGGTTTGGCTTGATAAACTGTTGAAATGAAAGTTCCATTTATTCAGAGATTAAAACTTCATCTTACTTTAAACCCTGTTCTTTATGATAAACCTAAACTTTTCAACGCTGGCTCACGTAAT
CAAATTTGAGGATGCAGCCATGGACAATTTCACTCGTGTGAAACTCCGTCTCTACTAAAAATAAAAAATGTGATTTATGTGATCCAGTACTGGGAATAAGAAATTTACCCAGCAGGTTCTCCACCTTGCTAGTGGGAACAGTGAGACTCCGTTAAAAAAACAAAAAAAAACG
TCT }
```

Sanchez Mendez Edmundo Josué
Análisis de Algoritmos