PS5 DNA Sequence Alignment

This task required us to compare two ASCII strings in order to determine their edit distance, as well as provide space and time analysis on the program while it was running. Using a dynamic programming method, we were required to create a program that computed an optimum sequence alignment on two DNA sequences. The space complexity of our program might then be measured using Valgrind, a programming tool.

Key concepts

The EDistance class represents a matrix of integers with a vector of vectors of type int. The function optDistance() would then use the min() and penalty() functions to fill the matrix from bottom to top with the smallest of three distances, then return the optimal distance between the two sequences. The alignment() method would then go backwards through the populated matrix, constructing the output alignment string depending on the best path.

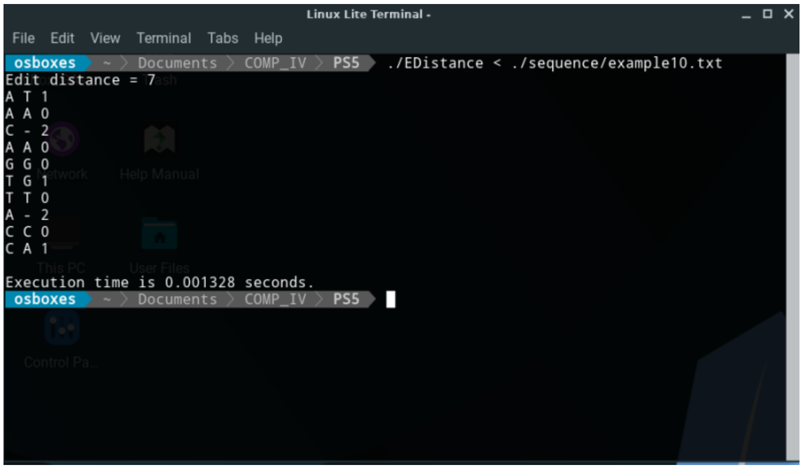
What I Accomplished

My EDistance algorithm uses dynamic programming to fill a matrix with each computation from bottom right to top left, culminating the ideal distance at [0][0]. Tracing the matrix in reverse order from top left to bottom right also shows the alignment path. Instead of computing the same subproblem numerous times, the dynamic programming technique allows the alignment computations to be divided down into subproblems and then stored.

What I Learned

Apart from the numerous applications of Edit Distance, I learned about the advantages of dynamic programming, which, when compared to the recursive solution for this assignment, would have had a much higher space complexity due to the number of recursive calls exceeding 2N when comparing two strings of length N. In addition to, I also gained the knowledge on how to use the Valgrind massif tool to efficiently monitor program memory use. This utility shows how much memory the heap consumed during execution. I was able to determine the projected run time and memory use of a bigger sample of strings of length N using the doubling approach.

Output



```
osboxes  ~ > Documents > COMP_IV > PS5  ./EDistance < ./sequence/example10.txt
Edit distance = 7
A T 1
A A 0
C - 2
A A 0
G G 0
T G 1
T T 0
A - 2
C C 0
C A 1

Execution time is 0.001328 seconds.
osboxes  ~ > Documents > COMP_IV > PS5
```

```
all:
	make EDistance

EDistance: main.o EDistance.o
	g++ -o EDistance -g main.o EDistance.o -lsfml-system

main.o: EDistance.cpp main.cpp
	g++ -c -g main.cpp -ansi -pedantic -Wall -Werror

EDistance.o: EDistance.hpp EDistance.cpp
	g++ -c -g -O2 EDistance.cpp -ansi -pedantic

clean:
	rm -f *.o *~ EDistance
```

```cpp
#include <iostream>
#include <cstring>
#include <SFML/System.hpp>

#include "EDistance.hpp"

using namespace std;

int main() {

  // Clock
  sf::Clock clock;
  sf::Time t;


  string String1;
  string String2;

  // Get input
  cin >> String1;
  cin >> String2;

  // Initialize my class, all math done in constructor and sets relevant
  // member variables
  EDistance output(String1, String2);


  // Get desired output
  cout << "Edit distance = " << output.getEditDistance() << endl;
  cout << output.getEditString() << endl;

  t = clock.getElapsedTime();
  cout << "Execution time is " << t.asSeconds() << " seconds." << endl;

  return 0;
}
```

```cpp
#ifndef EDistance_HPP
#define EDistance_HPP

#include <cstring>
#include <vector>

class EDistance {
public:
  EDistance(std::string _stringA, std::string _stringB);
  void printOpt() const;
  int getEditDistance() const;
  std::string getEditString() const;

private:
  // Input Variables
  std::string stringA;
  std::string stringB;

  // Constructed Variables
  std::vector< std::vector<int> > opt;
  int editDistance;
  std::string editString;

  // Private Functions
  int optDistance();
  std::string alignment() const;
  int penalty(char a, char b);
  int min(int a, int b, int c);
};

#endif
```

```
#include <iostream>
#include <cstring>
#include <vector>
#include <sstream>

#include "EDistance.hpp"

using namespace std;

EDistance::EDistance(string _stringA, string _stringB) : stringA(_stringA), stringB(_stringB) {

  vector<int> temp;

  // Populate the matrix with 0's to start
  for(unsigned i = 0; i < stringB.length() + 1; i++)
    temp.push_back(0);
  for(unsigned i = 0; i < stringA.length() + 1; i++)
    opt.push_back(temp);

  editDistance = optDistance();
  editString = alignment();
}

int EDistance::penalty(char a, char b) {

  if(a == b)
    return 0;
  else
    return 1;
}
int EDistance::min(int a, int b, int c) {

  if(a <= b && a <= c)
    return a;
  else if(b <= c)
    return b;
  else
    return c;
}

int EDistance::optDistance() {

  // Fill in the matrix with the EditDistances
  for(int i = opt.size() - 1; i >= 0; i--)
    for(int j = opt[i].size() - 1; j >= 0; j--) {
      if((i == opt.size() - 1) && (j == opt[i].size() - 1))
        opt[i][j] = 0;
      else if(i == opt.size() - 1)
        opt[i][j] = opt[i][j + 1] + 2;
      else if(j == opt[i].size() - 1)
        opt[i][j] = opt[i + 1][j] + 2;
      else
        opt[i][j] = min(opt[i+1][j+1] + penalty(stringA[i], stringB[j]),
                        opt[i+1][j] + 2,
                        opt[i][j+1] + 2);
    }

  return opt[0][0];
}
string EDistance::alignment() const {

  stringstream ss;

  unsigned i = 0, j = 0;

  while(i < opt.size() - 1 || j < opt[0].size() - 1) {
    if((i < opt.size() - 1)
       && (j < opt[0].size() - 1)
       && (opt[i+1][j+1] <= opt[i+1][j] + 1)
       && (opt[i+1][j+1] <= opt[i][j+1] + 1)) {
      ss << stringA[i] << " " << stringB[j] << " " << opt[i][j] - opt[i+1][j+1] << ’\n’;
      i++;
      j++;
    }
    else if(((i < opt.size() - 1) && (opt[i+1][j] <= opt[i][j+1]))
            || (j == opt[0].size() - 1)) {
      ss << stringA[i] << " " << "-" << " " << opt[i][j] - opt[i+1][j] << ’\n’;
      i++;
    }
    else {
      ss << "-" << " " << stringB[j] << " " << opt[i][j] - opt[i][j+1] << ’\n’;
      j++;
    }
  }

  return ss.str();
}

void EDistance::printOpt() const {

  // Print the Matrix
  for(unsigned i = 0; i < opt.size(); i++) {
    for(unsigned j = 0; j < opt[i].size(); j++) {
      cout.width(4);
      cout << opt[i][j];
    }
    cout << endl;
  }
}

int EDistance::getEditDistance() const {

  return editDistance;
}

string EDistance::getEditString() const {

  return editString;
}
```

```
/**********************************************************************
 *  readme
 *  DNA Sequence Alignment
 **********************************************************************/
```

Name: Matthew Lorette Anaya

Hours to complete assignment: 5

```
/**********************************************************************
 * Explain which approach you decided to use when implementing
 * (either recursive with memoization, recursive without memoization,
 * dynamic programming or Hirschberg\222s algorithm). Also describe why
 * you chose this approach and what its pros and cons are.
 **********************************************************************/
```

Implementation of this program was done with the use of dynamic programming and a matrix. I used the algorithm on the Princeton site in order to fill said matrix. In-order to find the alignment I used backtracking top-left to bottom right, moving from the current matrix index to the next-lowest matrix index. There was a certain case where if the diagonal was 1 higher than the downwards or rightwards option, diagonal was still the taken rout. In any case, depending on which direction I went, I either added a gap, or both letters, and incremented i and j counters to traverse back to the bottom right of the matrix.

```
/**********************************************************************
 * Does your code work correctly with the endgaps7.txt test file?
 *
 * This example should require you to insert a gap at the beginning
 * of the Y string and the end of the X string.
 **********************************************************************/
```

Kinda confused here on what this question really is. The pdf is using example10.txt And this is asking for endgaps7.txt. Seems like there is a mix up of pdfs between different years of this . So I’m going to use the what the HW pdf says as there really isn’t an example to compare to otherwise. Though it also says to put this all into a folder named ps3, which is definitely incorrect.

Input:

```
        Ê./EDistance < ./sequence/example10.txt
```

Expected output:

```
        Edit distance = 7
        AT1
        AA0
        C-2
        AA0
        GG0
        TG1
        TT0
        A-2
        CC0
        CA1
```

What happened:

```
        Edit distance = 7
        A T 1
        A A 0
        C - 2
        A A 0
        G G 0
        T G 1
        T T 0
        A - 2
        C C 0
        C A 1

        Execution time is 0.00094 seconds.
```

```
/**********************************************************************
 * Look at your computer\222s specs in the settings.
 * How much RAM does your computer have and explain what this means?
 **********************************************************************/
```

My Mac has 16gb of RAM. Random access memory gives applications a place to store and access data on a short-term basis. It stores the information your computer is actively using so that it can be accessed quickly.

```
/**********************************************************************
 *  For this question assume M=N. Look at your code and determine
 *  approximately how much memory it uses in bytes, as a function of
 *  N. Give an answer of the form a * N^b for some constants a
 *  and b, where b is an integer. Note chars are 2 bytes long, and
 *  ints are 4 bytes long.
 *
 *  Provide a brief explanation.
 *
 *  What is the largest N that your program can handle if it is
 *  limited to 8GB (billion bytes) of memory?
 **********************************************************************/
```

N^2 is the area of the matrix, the number of integer slots that need to be filled in. 4 is the size of an integer in bytes.

a = 4
b = 2
largest N = ˜44,721

Explanation:

        4 * 44,721^2 = 7,999,871,364 just shy of 8gb.

```
/**********************************************************************
 * Run valgrind if you can and attach the output file to your submission.

 * If you cannot run it, explain why, and list all errors you\222re seeing.
 * If you can run it successfully, does the memory usage nearly match that
 * found in the question above?
 * Explain why or why not.
```

```
/*********************************************************************

-------------------------------------------------------------------------------
  n       time(i)        total(B)   useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
 67  6,808,807,498    3,085,122,584    3,084,577,402       545,182            0
 68  6,870,407,882    3,146,681,624    3,146,125,562       556,062            0
 69 30,570,148,197    3,201,904,240    3,201,338,395       565,845            0

It does not, its actually quite different and I’m not entirely sure as to why. Not
Sure if I’m reading valgrind output wrong or my equation is.


/*********************************************************************
 *  For each data file, fill in the edit distance computed by your
 *  program and the amount of time it takes to compute it.
 *
 *  If you get segmentation fault when allocating memory for the last
 *  two test cases (N=20000 and N=28284), note this, and skip filling
 *  out the last rows of the table.
 *********************************************************************/

data file           distance       time (seconds)
-------------------------------------------------
ecoli2500.txt        118                     0.125216
ecoli5000.txt        160                     0.334861
ecoli7000.txt        194                     0.521017
ecoli10000.txt       223              1.4272
ecoli20000.txt       3135            74.6052
ecoli28284.txt       8394           177.645

/**********************************************************************
 *  Here are sample outputs from a run on a different machine for
 *  comparison.
 **********************************************************************/

data file           distance       time (seconds)
------------------------------------------------
ecoli2500.txt          118            0.171
ecoli5000.txt          160            0.529
ecoli7000.txt          194            0.990
ecoli10000.txt         223            1.972
ecoli20000.txt         3135           7.730



/*********************************************************************
 *  For this question assume M=N (which is true for the sample files
 *  above). By applying the doubling method to the data points that you
 *  obtained, estimate the running time of your program in seconds as a
 *  polynomial function a * N^b of N, where b is an integer.
 *  (If your data seems not to work, describe what went wrong and use
 *  the sample data instead.)
 *
 *  Provide a brief justification/explanation of how you applied the
 *  doubling method.
 *
 *  What is the largest N your program can handle if it is limited to 1
 *  day of computation? Assume you have as much main memory as you need.
 **********************************************************************/
a =
b =
largest N =

/
**********************************************************************
 *  Did you use the lambda expression in your assignment? If yes, where
 * (describe a method or provide a lines numbers)
 **********************************************************************/
No



**********************************************************************
 *  List whatever help (if any) you received from the course TAs,
 *  instructor, classmates, or anyone else.
 **********************************************************************/

N/a

/**********************************************************************
 *  Describe any serious problems you encountered.
 **********************************************************************/



/**********************************************************************
 *  List any other comments here.
 **********************************************************************/

```