

PS4 DNA Sequence Alignment

In this project, I had to use the Needleman-Wunsch method to find the best alignment for two strings. In this assignment OO designs that were implemented were classes, and data structures used were arrays to create a 2d array in order to implement the Needlema-Wunsch method. In this assignment I learned different ways to use valgrind in order to find memory leaks, and the SFML library in order to determine the run time for different sequences.

One problem that I ran into was that I got stuck on how to delete a 2d array to get rid of memory leaks, but to solve this problem I used google to help me with it.

```
adam@adam-XPS-13-9360: ~/Desktop/COMP4/PS4
File Edit View Search Terminal Help
adam@adam-XPS-13-9360:~/Desktop/COMP4/PS4$ ls
bothgaps20.txt  ED.cpp  ED.o      main.o    readme.txt  test.txt
ED              ED.hpp  main.cpp  Makefile  sequence
adam@adam-XPS-13-9360:~/Desktop/COMP4/PS4$ ./ED < bothgaps20.txt
Edit distance: 12
a a 0
- z 2
- z 2
b b 0
c c 0
d d 0
e e 0
f f 0
g g 0
h h 0
i i 0
z - 2
z - 2
z - 2
z - 2
j j 0
k k 0
l l 0
m m 0
n n 0
o o 0
p p 0

Execution time is 0.000147 seconds
adam@adam-XPS-13-9360:~/Desktop/COMP4/PS4$
```

```
1: CC = g++
2: CFLAGS = -std=c++11 -c -g -Wall -Werror -pedantic
3: LIBS = -lsfml-system
4:
5: all: ED
6:
7: ED: main.o ED.o
8:     $(CC) main.o -o ED $(LIBS)
9:
10: main.o: main.cpp
11:     $(CC) -c $(CFLAGS) main.cpp
12:
13: planets.o: ED.cpp ED.hpp
14:     $(CC) -c $(CFLAGS) ED.cpp ED.hpp
15:
16: clean:
17:     rm *.o ED massif*
```

```
1: #include <SFML/System.hpp>
2: #include <iostream>
3: #include <vector>
4: #include <string>
5:
6: using namespace std;
7: using namespace sf;
8:
9: class ED {
10: public:
11:     ED(string a, string b);
12:
13:     int penalty(char a, char b);
14:     int min(int a, int b, int c);
15:     int OptDistance();
16:     string Alignment();
17:     int getCost();
18:
19:     ~ED();
20:
21: private:
22:     string x, y;
23:     int M, N, cost;
24:     int** opt;
25:
26: };
```

```
1: #include "ED.hpp"
2:
3: ED::ED(string a, string b) {
4:
5:     this->x = a;
6:     this->y = b;
7:
8:     x += '-';
9:     y += '-';
10:
11:     M = x.length() + 1;
12:     N = y.length() + 1;
13:
14:     opt = new int*[M];
15:     for (int i = 0; i < M; i++)
16:         opt[i] = new int[N];
17: }
18:
19: int ED::penalty(char a, char b) {
20:     if (a == b)
21:         return 0;
22:     else
23:         return 1;
24: }
25:
26: int ED::min(int a, int b, int c) {
27:     if (a < b && a < c)
28:         return a;
29:     else if (b < a && b < c)
30:         return b;
31:     else
32:         return c;
33: }
34: int ED::OptDistance() {
35:
36:     //add bottom outside
37:     int j = 0;
38:     for (int i = N-1; i >= 0; i--) {
39:         opt[M-1][i] = j;
40:         j += 2;
41:     }
42:
43:     //add right outside
44:     j = 0;
45:     for (int i = M-1; i >= 0; i--) {
46:         opt[i][N-1] = j;
47:         j += 2;
48:     }
49:
50:     //compare
51:     for (int i = M-2; i >= 0; i--) {
52:         for (int j = N-2; j >= 0; j--) {
53:             if (x[i] != y[j]) {
54:                 opt[i][j] = min(opt[i+1][j] + 2, opt[i][j+1] + 2, opt[i+1][j
+1] + 1);
55:             }
56:             else
57:                 opt[i][j] = opt[i+1][j+1];
58:         }
59:     }
60:
```

```
61:     string out = Alignment();
62:     cout << "Edit distance: " << cost << endl;
63:     cout << out << endl;
64:
65:     for (int i = 0; i < M; i++)
66:         delete [] opt[i];
67:
68:     delete [] opt;
69:     return 0;
70: }
71: string ED::Alignment() {
72:     vector<char> out;
73:
74:     int i = 0,
75:         j = 0,
76:         pen;;
77:     while (i < M-2 || j < N-2) {
78:         if (x[i] == y[j]) {
79:             pen = penalty(x[i], y[j]);
80:             out.push_back(x[i]);
81:             out.push_back(' ');
82:             out.push_back(y[j]);
83:             out.push_back(' ');
84:             i++;
85:             j++;
86:         }
87:         else if (opt[i][j] == opt[i+1][j+1] + 1) {
88:             pen = penalty(x[i], y[j]);
89:             out.push_back(x[i]);
90:             out.push_back(' ');
91:             out.push_back(y[j]);
92:             out.push_back(' ');
93:             i++;
94:             j++;
95:         }
96:         else if (opt[i][j] == opt[i+1][j] + 2) {
97:             pen = penalty(x[i], y[j]) + 1;
98:             out.push_back(x[i]);
99:             out.push_back(' ');
100:             out.push_back('-');
101:             out.push_back(' ');
102:             i++;
103:         }
104:         else if (opt[i][j] == opt[i][j+1] + 2) {
105:             pen = penalty(x[i], y[j]) + 1;
106:             out.push_back('-');
107:             out.push_back(' ');
108:             out.push_back(y[j]);
109:             out.push_back(' ');
110:             j++;
111:         }
112:         out.push_back('0' + pen);
113:         cost += pen;
114:         out.push_back('\n');
115:     }
116:     string new_out(out.begin(), out.end());
117:     return new_out;
118: }
119:
120: int ED::getCost() {
121:     return cost;
```

```
122: }  
123:  
124: ED::~~ED() {}
```

```
1: #include "ED.cpp"
2:
3: int main(int argc, char* argv[]) {
4:
5:     string a, b;
6:     cin >> a;
7:     cin >> b;
8:
9:     ED ed(a, b);
10:
11:     Clock clock;
12:     Time t;
13:     ed.OptDistance();
14:     t = clock.getElapsedTime();
15:     cout << "Execution time is " << t.asSeconds() << " seconds\n";
16:     //cout << "Edit distance (for longer sequences): " << ed.getCost() << en
dl;
17:
18:     return 0;
19: }
```