

```
1: #include <iostream>
2: #include <cstring>
3: #include <vector>
4: #include <sstream>
5:
6: #include "EDistance.hpp"
7:
8: using namespace std;
9:
10: EDistance::EDistance(string _stringA, string _stringB) : stringA(_stringA
), stringB(_stringB) {
11:
12:     vector<int> temp;
13:
14:     // Populate the matrix with 0's to start
15:     for(unsigned i = 0; i < stringB.length() + 1; i++)
16:         temp.push_back(0);
17:     for(unsigned i = 0; i < stringA.length() + 1; i++)
18:         opt.push_back(temp);
19:
20:     editDistance = optDistance();
21:     editString = alignment();
22: }
23:
24: int EDistance::penalty(char a, char b) {
25:
26:     if(a == b)
27:         return 0;
28:     else
29:         return 1;
30: }
31: int EDistance::min(int a, int b, int c) {
32:
33:     if(a <= b && a <= c)
34:         return a;
35:     else if(b <= c)
36:         return b;
37:     else
38:         return c;
39: }
40:
41: int EDistance::optDistance() {
42:
43:     // Fill in the matrix with the EditDistances
44:     for(int i = opt.size() - 1; i >= 0; i--)
45:         for(int j = opt[i].size() - 1; j >= 0; j--) {
46:             if((i == opt.size() - 1) && (j == opt[i].size() - 1))
47:                 opt[i][j] = 0;
48:             else if(i == opt.size() - 1)
49:                 opt[i][j] = opt[i][j + 1] + 2;
50:             else if(j == opt[i].size() - 1)
51:                 opt[i][j] = opt[i + 1][j] + 2;
52:             else
53:                 opt[i][j] = min(opt[i + 1][j + 1] + penalty(stringA[i], stringB[j]),
54:                                opt[i + 1][j] + 2,
55:                                opt[i][j + 1] + 2);
56:         }
57:
58:     return opt[0][0];
59: }
60: string EDistance::alignment() const {
61:
62:     stringstream ss;
63:
64:     unsigned i = 0, j = 0;
```

```
65:
66:     while(i < opt.size() - 1 || j < opt[0].size() - 1) {
67:         if((i < opt.size() - 1)
68:            && (j < opt[0].size() - 1)
69:            && (opt[i+1][j+1] <= opt[i+1][j] + 1)
70:            && (opt[i+1][j+1] <= opt[i][j+1] + 1)) {
71:             ss << stringA[i] << " " << stringB[j] << " " << opt[i][j] - opt[i+1
][j+1] << '\n';
72:             i++;
73:             j++;
74:         }
75:         else if(((i < opt.size() - 1) && (opt[i+1][j] <= opt[i][j+1]))
76:                || (j == opt[0].size() - 1)) {
77:             ss << stringA[i] << " " << "-" << " " << opt[i][j] - opt[i+1][j] <<
'\n';
78:             i++;
79:         }
80:         else {
81:             ss << "-" << " " << stringB[j] << " " << opt[i][j] - opt[i][j+1] <<
'\n';
82:             j++;
83:         }
84:     }
85:
86:     return ss.str();
87: }
88:
89: void EDistance::printOpt() const {
90:
91:     // Print the Matrix
92:     for(unsigned i = 0; i < opt.size(); i++) {
93:         for(unsigned j = 0; j < opt[i].size(); j++) {
94:             cout.width(4);
95:             cout << opt[i][j];
96:         }
97:         cout << endl;
98:     }
99: }
100:
101: int EDistance::getEditDistance() const {
102:
103:     return editDistance;
104: }
105:
106: string EDistance::getEditString() const {
107:
108:     return editString;
109: }
```