

```
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() {}
```