

Homework 2

Elliott Pryor

7 September 2021

Problem 1

Propose a dynamic programming algorithm to solve longest common substring problem

Problem 2 Given two sequences S, T (not necessarily the same length), let G, L, H be the scores of the optimal global alignment, optimal local alignment, and optimal global alignment without penalizing leading or trailing spaces.

- a) Give an example of S, T so that the three scores are different
- b) Prove or disprove the statement $L \geq H \geq G$

Problem 3

Implement the global alignment algorithm from class. Your program should read in a FASTA file (see HW1). You can assume that the file just contains two sequences, e.g.

```
> seq1 ACTGGGAAA > seq2 CTGGAACA
```

The filename should be supplied as a command-line parameter.

Align the first string with the second string. Print out one optimal alignment.

You can assume a simplified scoring function δ that has the following form: $\delta(\text{match}) = 2$ $\delta(\text{mismatch}) = -1$ $\delta(\text{insertion/deletion}) = -1$

Demonstrate your algorithms on two test cases (use screen shots to show runs).

```
1 import sys
2 import numpy as np
3
4
5 def delta(c1, c2):
6     # Helper function used for scoring
7     if c1 == c2:
8         return 2
9     else:
10        return -1
11
12
13 def construct_alignment(P, S, T, i=-1, j=-1, out_S='', out_T=''):
14     # Reconstruct the optimal alignment from the output of needleman(S,T)
15     if i == -1 and j == -1:
16         i, j = len(S), len(T)
17
18     if i == 0 and j == 0: # base case
19         print('*' * 80) # print out the optimal alignment
20         print(out_S)
21         print(out_T)
22         return out_S, out_T
23
24     retS, retT = '', ''
25     moves = P[i][j]
26     for move in moves:
27         if move == 0: # is just a part of the output of P, due to its construction. So
            skip move
28             continue
29         elif move == 1: # diagonal
30             out_S = S[i - 1] + out_S
31             out_T = T[j - 1] + out_T
32             i = i - 1
33             j = j - 1
34         elif move == 2: # delete
35             out_S = S[i - 1] + out_S
36             out_T = '_' + out_T
```

```

37         i = i - 1
38         j = j
39     elif move == 3: # insert
40         out_S = '_' + out_S
41         out_T = T[j-1] + out_T
42         i = i
43         j = j - 1
44     retS, retT = construct_alignment(P, S, T, i, j, out_S, out_T) # recur
45
46     return retS, retT # return the last optimal alignment
47
48
49 def needleman(S, T):
50     V = np.zeros((len(S) + 1, len(T) + 1)) # value
51     P = np.zeros((len(S) + 1, len(T) + 1)) # path
52
53     # initialize V
54     V[:, 0] = [-i for i in range(len(S) + 1)]
55     V[0, :] = [-i for i in range(len(T) + 1)]
56     P[:, 0] = 2 * np.ones(len(S) + 1) # vertical case 2
57     P[0, :] = 3 * np.ones(len(T) + 1) # horizontal case 1
58     P[0, 0] = 0 # reset 0, 0
59
60     P = [[x] for x in row] for row in P # convert to list of lists
61     print(P)
62     for r in range(1, len(V)): # 2nd row onwards
63         for c in range(1, len(V[r])): # 2nd col onwards
64
65             replace = V[r-1, c-1] + delta(S[r-1], T[c-1])
66             delete = V[r-1, c] + delta(S[r-1], '_')
67             insert = V[r, c - 1] + delta('_', T[c-1])
68
69             arr = np.array([replace, delete, insert])
70
71             V[r, c] = max(arr)
72             if replace == max(arr):
73                 P[r][c].append(1)
74             if delete == max(arr):
75                 P[r][c].append(2)
76             if insert == max(arr):
77                 P[r][c].append(3)
78
79     return V, P
80
81
82 def read_fasta(filepath):
83     """
84     Returns a dict key=sequence_name, value=sequence from the fasta file
85     :param filepath:
86     :return:
87     """
88     file = open(filepath)
89     print(f"Opening: {filepath}")
90     started = False
91     header = ''

```

```
92     text = ''
93     sequences = {}
94     for line in file:
95         if line.startswith('>'):
96             if started:
97                 sequences[header] = line.replace('\n', '')
98                 text = ''
99             started = True
100             header = line.replace('>', '').strip()
101         else:
102             text += line.strip()
103     sequences[header] = line.replace('\n', '')
104     return sequences
105
106
107 if __name__ == '__main__':
108     if len(sys.argv) == 2:
109         # run fasta file
110         # expect arguments z_algorithm.py <path to file>
111         filepath = sys.argv[1]
112         seq = read_fasta(filepath)
113         vals = list(seq.values())
114         keys = list(seq.keys())
115         print(f"Computing alignment between S={keys[0]} and T={keys[1]}")
116         V, P = needleman(vals[0], vals[1])
117         print(f"Optimal Score: {V[-1,-1]}")
118         s, t = construct_alignment(P, vals[0], vals[1])
119     else:
120         V, P = needleman('cat', 'taat')
121         print('*' * 80)
122         print(V)
123         print()
124         print(P)
125         print(f"Optimal Score: {V[-1, -1]}")
126         print('*' * 80)
127
128         s, t = construct_alignment(P, 'cat', 'taat')
129         print(s)
130         print(t)
131         pass
```