

**BCB/CprE/ComS 548 Fundamental Algorithms in Computational Biology**  
**Fall 2007**  
**Homework 1**  
**Due Tuesday, September 11**

1. (5 points) Find the best local alignment(s) between the two sequences *AGCCAT* and *GTGCCTGA*. Use the following scoring system: 4 for a match,  $-3$  for a mismatch and  $-2$  for a gap.
2. (5 points) The purpose of this problem is to experiment with sequence alignment software. Point your browser to the URL <http://deepc2.psi.iastate.edu/aat/align/align.html> which provides a variety of alignment software (developed by ISU professor Xiaoqiu Huang). The interface uses the FASTA format, in which a sequence such as AGGCAT is represented as

```
>name
AGGCAT
```

where the sequence name is optional. Run the global alignment program GAP and local alignment program SIM on the two *E. Coli* genomic sequences from the text file accompanying the homework (found on the course web under “homeworks  $\rightarrow$  homework 1 supplement”). Try the default parameter values. Next, use the parameters: mismatch penalty = 10, gap opening penalty = 0, gap extension penalty = 5; Then, use the same parameters except for increasing the gap opening penalty to 10. Try other parameter values as you see fit and comment on the results.

3. (5 points) Consider local alignment. Suppose that instead of the being interested in the entire alignment path, we are interested in knowing only the aligning substrings. If the optimal alignment path starts at  $S[i, j]$  and continues to  $S[r, t]$ , then the substring  $\mathcal{A}[i, r - 1]$  is aligned to substring  $\mathcal{B}[j, t - 1]$ . Describe a simple algorithm that finds these substrings while using  $O(n + m)$  space and running in  $O(nm)$  time. This is equivalent to finding the endpoint of the optimal path and the start point of the optimal path. You can assume for simplicity that these endpoints are unique. Your algorithm should not involve dividing the dynamic programming table into parts.
4. (10 points) In discussing the space saving technique, we chose to represent the alignment path as an *intersection list*. Each intersection is a pair of integers that describes the movement between adjacent rows in the table. The entire alignment path is defined by  $n$  intersections. Alternatively, we could represent the alignment path using a *path string*, a string using the alphabet  $\{D, V, H\}$ , where each character corresponds to a move along the path.  $D$  corresponds to a diagonal move,  $V$  corresponds to a vertical move, and  $H$  corresponds to a horizontal move.
  - (a) True or False: “The alignment length  $L$  is the same as the length of the path string.”
  - (b) Describe how to construct the path string give an intersection list and the lengths of the two strings:  $n$  and  $m$ .
  - (c) Describe how to construct  $\mathcal{A}_A$  and  $\mathcal{B}_A$  given a path string and two strings:  $\mathcal{A}$  and  $\mathcal{B}$ .
  - (d) Assume  $n = m$ . Each character in the path string can be efficiently stored using 2 bits. Each intersection in the interval list can be stored using  $\lceil \log_2(n) \rceil + 1$  bits. Describe, as a function of  $n$ , the number of bits needed to store the shortest possible path string. Describe the number of bits needed to store the longest possible path string. Describe the number of bits needed to store the intersection list.

- (e) Describe at least one positive of using the intersection list as the representation of the path. Describe at least one positive of using the path string.
5. (5 points) In the computer science literature, the terms *substring* and *subsequence* have different meanings. The characters in a substring of a string  $\mathcal{A}$  must occur contiguously in  $\mathcal{A}$ , whereas characters in a subsequence might be interspersed with characters not in the subsequence. For example,  $bbz$  is a subsequence of  $abxyabxz$ . Given two sequences  $\mathcal{A}$  and  $\mathcal{B}$ , a common subsequence is a subsequence that appears in both  $\mathcal{A}$  and  $\mathcal{B}$ . The longest common subsequence (LCS) problem is to find a longest subsequence of  $\mathcal{A}$  and  $\mathcal{B}$ . Describe a choice of parameters  $\alpha$ ,  $\beta$ , and  $\gamma$  that results in a longest common subsequence between  $\mathcal{A}$  and  $\mathcal{B}$  corresponding exactly to the matching characters in  $\mathcal{A}_A$  and  $\mathcal{B}_A$  after doing global alignment. Explain why this is the case. Can there be more than one longest common subsequence?