CS900 Lab - Day 14

July 17th, 2003

Instructor: Thomas Hofmann

New Topics Covered
String matching, alignment, Lecture: Day 14

Installing
Copy the file align.cpp from the course directory

```
cp /course/cs900/day14/src/align.cpp .
```

Compiling
You should use the gnu C++ compiler g++. Your program will be in align.cpp . To compile type

```
g++ align.cpp -o align -Wall -O4
```

Problem 1: Alignment Cost Matrix

Implement a program that computes the alignment cost matrix for two strings, using the recurrence relations discussed in class. A file with six protein sequences has been provided under

```
/course/cs900/day14/data/hemoglobin.txt
```

These are 6 hemoglobin sequences taken from the SWISSPROT database. The provided align.cpp program already contains code that reads in the sequences and ask the user to select which sequences to compare. The program will thus start like this

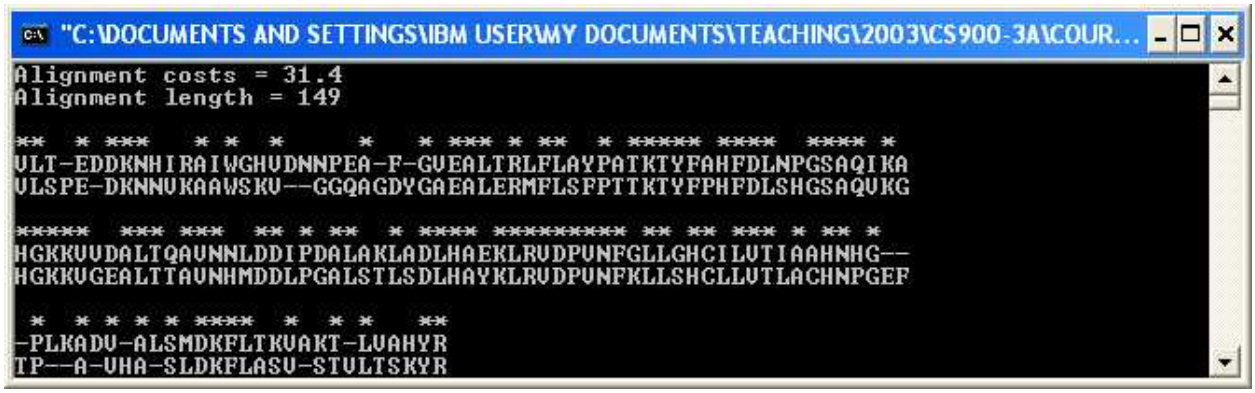
Once the two strings to compare have been selected, an appropriate two-dimensional array to store the values of the alignment cost matrix (A) gets created. This happens in the lines

```
double **A  = new double*[len1+1];
for (n=0; n<=len1; n++) A[n] = new double[len2+1];
```

After this, you can refer to elements A[k][l], where k,l are within the valid range. Notice that an index of 0 corresponds to an empty string prefix (compare this with the table shown in class). Apply the recurrence relations and fill the table with value from left to right and from top to bottom until you have computed the final value A[len1,][len2]. Output this value as the score of the optimal alignment. It will in general be higher for "more similar" sequences.

Problem 2: Optimal Alignment

Use the table computed in the two-dimensional array A to reconstruct an optimal alignment. To do this you have to trace back, where the value you computed "came from". Starting in the lower right corner, you will check whether the maximal value was obtained via a horizontal, diagonal, or vertical "move". Horizontal means that there is a gap introduced in the alignment of string2, vertical that there is a gap introduced in the alignment of string1 and diagonal that the two positions have been aligned with one another. Output the alignment. You may also keep track of positions that are correctly matched up.



You may also consider modifying the code written for Problem 1 in a way that the information from where the maximum came (up,left,diagonal) is stored in an appropriate two-dimensional array. Then, tracing back the optimal alignment becomes somewhat easier.