# CS590 homework 4 – Dynamic Programming, Greedy Algorithms

The due date for this assignment is Sunday, April 17th, at 11.59pm. This assignment is worth 10% of your final grade.

Any sign of collaboration will result in a 0 and being reported to the Graduate Academic Integrity Board. The programming assignment will be done individually. No collaboration is allowed between students. No code from online resources is allowed to be used besides the code that I will share with you. Late submission policy described in the syllabus will be applied.

The class **random_generator** has been updated (random_generator.h and random_generator.cc) by a member function which generates random strings of a fixed length using the a given number of characters from the alphabet, starting with "a".

- `char* random_string_m(int n, int no_ch)`
  The function allocates $n+1$ characters. The first $n$ characters $(0 \ldots n-1)$ are chosen at random using the first *no_ch* characters from the alphabet starting with "a" (e.g., for $no\_ch = 4$ the characters are randomly chosen out of {"a","b","c","d"}). The $n$-th character is set to 0 in order to mark the end of the string.

## Dynamic programming (70 points)

The dynamic programming Smith-Waterman algorithm is matching sequences recursively defined as follows, given $X = x_1, \ldots, x_n$ (along table rows) and $Y = y_1, \ldots, y_m$ (along table columns).

$$M(i,0) = 0, \text{ for all } 0 \le i \le n$$
$$M(0,j) = 0, \text{ for all } 0 \le j \le m$$
$$M(i,j) = \max \begin{cases} M(i-1,j-1) + 2 \text{ if } x_i = y_j \\ M(i-1,j-1) - 1 \text{ if } x_i \neq y_j \\ M(i-1,j) - 1 \text{ if "-" is inserted into } Y \\ M(i,j-1) - 1 \text{ if "-" is inserted into } X \end{cases}$$

The function $M(i,j)$ defines a so called matching score for the partial sequences $X_i$ and $Y_j$. If in the recursive definition of $M$ the maximum value is due to the third or fourth line, you have to insert the character "-" into either $X$ or $Y$ in order to reconstruct the matching sequences $X'$ and $Y'$. Similar to the LCS problem we need only need a table to store the $M(i,j)$ values, but an additional table that allows us to later generate $X'$ and $Y'$ from $X$ and $Y$.

| | **Example 1** | **Example 2** | **Example 3** |
|---|---|---|---|
| **X** | abababda | cacacccbab | cdbaabbdca |
| **X'** | a–bababda | ca–cacccbab | c–dba––abbdca |
| **Y'** | acbabab–a | cadaadcc––– | cadcacca–bd–– |
| **Y** | acbababa | bccadaadcc | cadcaccabd |
| **M(n,m)** | **12** | **4** | **5** |

| | **Example 4** | **Example 5** |
|---|---|---|
| **X** | caacbdacca | dcacccbbba |
| **X'** | caacb–dacc–a | dcacccb–bba |
| **Y'** | c––cbcd–ccba | dca–––badba |
| **Y** | bccbcdccba | aadcabadba |
| **M(n,m)** | **9** | **7** |

1. Implement the bottom-up version of the Smith-Waterman algorithm given by the recursive definition of the function $M$ (as seen on the slides).
2. Implement the top-down with memoization version of the Smith-Waterman algorithm given by the recursive definition of the function $M$.

   **Notes:**

   - How do you initialize the necessary tables given the definition of $M$. Keep in mind that you have to able to determine whether or not you already computed a table value (memoization).
   - Values could be negative, but is there a limit for how small they can get?
3. Implement the function `void PRINT-SEQ-ALIGN-X(...)` that takes a number of parameters and then recursively prints the matching sequence that is derived from $X$. Implement a separate function `void PRINT-SEQ-ALIGN-Y(...)` that recursively prints the matching sequence that is derived from $Y$.
4. Find the maximum alignment for $X =$ `dcdcbacbbb` and $Y =$ `acdccabdbb` by using the Smith-Waterman algorithm (see slides). Execute the pseudocode algorithm and fill the necessary tables $H$ and $P$ in a bottom-up fassion. Reconstruct the strings $X'$ and $Y'$ using the tables $H$ and $P$.
   **(7+20+8+15 = 50 points)**

**Exercise 15.1-2** Show, by means of a counterexample, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the **density** of a rod of length $i$ to be $\frac{p_i}{i}$, that is, its value per inch. The greedy strategy for a rod of length $n$ cuts off a first piece of length $i$, where $1 \leq i \leq n$, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length $n - i$.
**(7 points)**

**Exercise 15.1-5** The Fibonacci numbers are defines by recurrence (3.22). Give an $O(n)$ time dynamic-programming algorithm to compute the $n$-th Fibonacci number. Draw the subproblem graph. How many vertices and edges are in the graph?
**(8 points)**

**Exercise 15.4-1** Determine an LCS of $\langle 1, 0, 0, 1, 0, 1, 0, 1 \rangle$ and $\langle 0, 1, 0, 1, 1, 0, 1, 1, 0 \rangle$.
**(5 points)**

Remarks:

- You are not allowed to use code from online resources. Your submission will be tested against that, and will receive a 0, and a report to the Graduate Academic Integrity Board if it is detected.
- Your report has to be typed, and submitted in a pdf file.
- No additional libraries are allowed to be used
- A Makefile is provided for both problems to build the code in the Virtual Box.
- Your code has to compile, and will be graded on the Virtual Box.
- The programming, and testing will take some time. Start early.
- Feel free to use the provided source code for your implementation. You have to document your code.