# CSci 421
Introduction to Algorithms

Homework Assignment 3
Due: Tuesday, 13 Jul 2004

Summer 2004 Handout 3

W. L. Ruzzo July 6, 2004

**Reading Assignment:**
Read Chapter 5, 6.6, 6.8, 6.11.1, 7.1, 7.3.

**Homework:**

1. Simulate the Maximum Consecutive Subsequence algorithm on the following sequence.

   1, 2, -2, 1, 1, 2, -6, 1, 3, 10

   Show the values of `Suffix_Max` and `Global_Max` after each iteration through the main loop, as well as showing the starting and ending indices of the subsequences to which they implicitly refer.

2. Run the Knapsack algorithm (Fig 5.10, pg 110) on the sequence of weights $k_1 = 5$, $k_2 = 2$, $k_3 = 4$, $k_4 = 3$, and $k_5 = 6$, with knapsack capacity $K = 16$. Show a table like Fig 5.11 to summarize the computation. (Note: although Fig 5.11 shows only one symbol, some cells may contain both "I" and "O"; e.g., row $k_3 = 5$, column 5 in Fig 5.11 could be labeled "I/O".)

3. Give an algorithm to solve the following variant of the Knapsack Problem: In addition to the knapsack capacity $K$ and the weights (also called sizes) $k_i$ of the $n$ objects that may be placed in the knapsack, suppose you are also given *values* for each, i.e., $n$ positive real numbers $v_i, 1 \leq i \leq n$, and the goal is select a subset of the items so that (1) their total weight does not exceed the knapsack capacity, and (2) their total value is as large as possible subject to (1). I.e., find a set $I \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in I} v_i$ is maximized subject to the constraint that $\sum_{i \in I} k_i \leq K$. Briefly explain why your algorithm is correct. In particular, state carefully the "induction hypothesis" that characterizes it. Analyze its running time. (Note total weight $\leq K$ rather than $= K$, and that your output should be an optimal subset, not just its total weight or value. Also, note that the maximum value is unique, although there may be many subsets attaining that value; reporting any such subset is sufficient.)

4. Given two sorted lists of numbers $x_1 < x_2 < \cdots < x_n$ and $y_1 < y_2 < \cdots < y_m$, and a number $Z$, give an algorithm to find the set

   $$\{(i, j) \mid 1 \leq i \leq n; 1 \leq j \leq m \text{ such that } x_i + y_j = Z\}.$$

   Analyze its running time. Time $O(n + m)$ is possible.

5. 6.64.

6. Run the string alignment algorithm given in lecture (similar to Fig 6.27, pg 158) on strings $S = tcatag$ and $T = tataag$. Build the cost matrix and traceback pointers as in the example given in lecture. Assume that aligning two identical letters gives a score of $+2$, whereas aligning a letter with a mismatched letter or a gap ("–") gives a score of $-1$.

7. For many applications of the string alignment algorithm, including most biological applications, it's not true that "all gaps are created equal." For example, if a gap in an alignment reflects a rare evolutionary event where there was an insertion or deletion in one sequence with respect to the other, then 10 separate gaps of length 1 may be much less likely that one gap of length 10. E.g. abcdewxyz is more likely to be related to abcde0123456789wxyz than to 0a1b2c3d4e5w5x7y8z9.

   Extend the string alignment algorithm to handle either of the following gap cost models. (You may choose which of these options you want to do; extra credit for doing both.)

   (a) **General gaps costs:** For general gap costs, part of your input is a table defining a cost function $C(j), 1 \leq j \leq max(m, n)$ which gives the cost of a gap of length $j$. You may assume $C(j) < 0$ for all $j$.

   (b) **"Affine" gap costs:** For affine gap costs, you are given two negative constants $C_i$ and $C_e$, and a gap of length $j$ costs $C_i + C_e * j$. Typically, $C_i \ll C_e < 0$, so that it is fairly expensive to initiate a gap ($C_i$), and less expensize (per letter) to extend one ($C_e$).

   As usual, give the algorithm, explain why it is correct, and analyze its running time. [The fastest known algorithm for general gap costs is slower than the basic $O(mn)$ algorithm presented in lecture, but there is a version handling affine gap costs in that time.]