Week 11 (Q4 – Q7)

4. Construct an example with only three or four matrices where the worst multiplication order does at least 100 times as many element-wise multiplications as the best order.

5. Suppose the dimensions of the matrices *A*, *B*, *C*, and *D* are 20x2, 2x15, 15x40, and 40x4, respectively, and we want to know how best to compute *A*x*B*x*C*x*D*. Show the arrays **cost** and **last** computed by Algorithms matrixOrder() in the lecture notes.

6. We have a knapsack of size 10 and 4 objects. The sizes and the profits of the objects are given by the table below. Find a subset of the objects that fits in the knapsack that maximizes the total profit by the dynamic programming algorithm in the lecture notes.

p	10	40	30	50
s	5	4	6	3

7. *S1* is a sequence of *n1* characters and *S2* is a sequence of *n2* characters. All characters are from the set {'a', 'c', 'g', 't'}. An alignment is defined by inserting any number of character '_' (the underscore character) into *S1* and *S2* so that the resulting sequences *S1'*and *S2*' are of equal length. Each character in *S1'* has to be aligned with the same character or an underscore in the same position in *S2*' and vice versa. The cost of an alignment of *S1* and *S2* is defined as the number of underscore characters inserted in *S1* and *S2*. For example, *S1* = "ctatg" and *S2* = "ttaagc". One possible alignment is

   S1' = "ct_at_g_" and
   S2' = "_tta_agc"

   Both *S1'* and *S2'* have length 8 and the cost is 5. We want to find the minimum cost of aligning two sequences, denoted as alignment(*n1*, *n2*).

   (a) Give a recursive definition of alignment(*n1*, *n2*).
   (b) Draw the subproblem graph for alignment(3, 4).
   (c) Design a dynamic programming algorithm of alignment(*n1*, *n2*) using the bottom-up approach.