1. The dynamic programming algorithm presented for optimizing a sequence of matrix multiplications computes only the smallest number $m_{1n}$ of MUL operations required. Extend the solution so that it also outputs an optimal ordering for the matrix multiplications using parentheses, for example as follows:

$$(M_1 \times (M_2 \times M_3)) \times M_4$$

(Hint: Record the locations of `$\times$' operations that lead to the optimum, and then use them and "divide-and-conquer" to print an optimal parenthesizing.) Analyze the complexity of this extended algorithm.

2. Extend our dynamic programming solution for the Knapsack problem with a procedure which, after tabulating the values, traces back and outputs the contents of an optimal load (as indices of its items). Argue that this does not increase the asymptotic $O(nT)$ complexity of the solution.

3. (a) Draw a decision tree for executions of insertion sort, when the keys to be sorted are given in array $A[1..3] = [a, b, c]$. Label each leave with the corresponding sorted order produced by the algorithm.

(b) Consider a decision tree for insertion sort, when there are $n$ keys to be sorted. Sketch the structure of the tree: How many leaves are there? Estimate the largest, the smallest, and the average depth of the leaves of the tree.

4. Present a random access machine (RAM) that reads a binary number from the input tape and leaves its value in register $R_1$. (E.g., with input 1011 we want to get the value $8 + 2 + 1 = 11$ in register $R_1$.) Analyze the time and space complexity (a) according to the unit-cost model exactly and (b) according to the logarithmic-cost model using asymptotic $\Theta()$ notation.

5. Present a simple random access machine that receives a non-negative integer $n$ in register $R_1$ and computes the value $2^n$ to register $R_2$. (A brute-force algorithm is OK for this assignment.) Analyze the time and space complexity according to both the unit-cost and the logarithmic-cost model, using asymptotic $O()$ or $\Theta()$ notation.