ECS 122A - HW4 Solutions

1. 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 p_i/i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first of length i, where $1 \leq i \leq n$. It then continues by applying the greedy strategy to the remaining piece of length $n - i$.

   Solution (4pts): Consider the following prices schedule for rod segments: \$120 for a rod of length 6, \$1 for a rod of length 4 and \$80 for a rod of length 5. Then, given a rod of length 10, the greedy method will first cut a rod of length 6, which has density \$20/inch, and then be forced to accept a rod of length 4, earning a total of \$121. However, cutting the rod into two rods of length 5 will earn a total of \$160, even though a rod of length 5 has density \$16/inch.

2. (15-4) **Printing neatly.** Consider the problem of neatly printing a paragraph with a monospaced font on a printer. The input text is a sequence of n words of length $l_1, l_2, \ldots, l_n$, measured in character. We want to print this paragraph neatly on a number of lines that hold a maximum of M characters each. Our criterion of "neatness" is as follows. If a given line contains words i through j, where $i \leq j$, and we leave exactly one space between words, the number of extra space characters at the end of the line is $M - j + i - \sum_{k=1}^{j} l_k$, which must be nonnegative so that the words fit on the line. We wish to minimize the sum, over all lines except the last, of the cubes of the numbers of extra space characters at the ends of lines. Give a dynamic programming algorithm to print a paragraph of n words neatly on a printer. Analyze the running time and space requirements of your algorithm.

   Solution (8 pts): We notice that the words we place onto a line depend on which words we decided to place onto all previous lines. Therefore, we create a table $T[j]$ to record the minimum penalty accrued by printing words $1 - j$ on the page, where word j is the last word of the document. We know that a document with zero words will have no penalties; therefore $T[0] = 0$. We then write the relationship between $T[j]$ and previous values as follows:

$$T[j] = \begin{cases} \min\limits_{1 \leq i < j} T[i-1] + (M - j + i - \sum\limits_{k=i}^{j} l_k)^3 & \text{if } \sum\limits_{k=i}^{j} l_k \leq M \text{, words } i \text{ through } j \text{ fit on the line} \\ \min\limits_{1 \leq i < j} T[i-1] & \text{if } \sum\limits_{k=i}^{j} l_k \leq M \text{ and } j = n\text{, this is the last line} \end{cases}$$

   Then, the best printing of all n input words will be $T[n]$. The table has an entry for each word and therefore has space complexity $O(n)$. Computing the minimum value for each word requires computing the minimum over the previous words; therefore, this algorithm runs in time $O(n^2)$.

3. LCS for input Sequences “AGGTAB” and “GXTXAYB”, provide thesolution matrix and the traceback.

Solution:

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \text{ (initials)} \\ c[i-1,j-1]+1 & \text{if } x[i] = y[j] \quad \text{(Case 1)} \\ \max\{c[i,j-1], c[i-1,j]\} & \text{if } x[i] \neq y[j] \quad \text{(Case 2)} \end{cases}$$

	y_j	A	G	G	T	A	B
x_i	0	0	0	0	0	0	0
G	0	↑ 0	↖ 1	↖ 1	← 1	← 1	← 1
X	0	↑ 0	↑ 1	↑ 1	↑ 1	↑ 1	↑ 1
T	0	↑ 0	↑ 1	↑ 1	↖ 2	← 2	← 2
X	0	↑ 0	↑ 1	↑ 1	↑ 2	↑ 2	↑ 2
A	0	↖ 1	↑ 1	↑ 1	↑ 2	↖ 3	← 3
Y	0	↑ 1	↑ 1	↑ 1	↑ 2	↑ 3	↑ 3
B	0	↑ 1	↑ 1	↑ 1	↑ 2	↑ 3	↖ 4

LCS: **GTAB**