CS180 Homework 5

Due: 8:00pm, 2/14/2019

1. Consider the problem of printing a paragraph with a mono-spaced font (all characters having the same width) on a printer. The input text is a sequence of n words of lengths $l_1, l_2, ..., l_n$, measured in characters. We want to print this paragraph neatly on a number of lines that hold a maximum of M characters each. 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 $M - j + i - \sum_{k=i}^{j} l_k$, which must be non-negative so that the words fit on the line. We wish to minimize the sum, over all lines except the last, of the *cube* of sum of the numbers of extra space characters at the ends of lines. Give an algorithm to print a paragraph of n words on a printer in a way that minimizes the above sum. Analyze the running time and space requirements of your algorithm.

2. The longest ascending subsequence problem is defined as follows. Given an array $A[1..n]$ of natural numbers, find the length of the longest ascending subsequence of A. (A subsequence is a list $A[i_1], A[i_2], ..., A[i_m]$ for some $1 \leq i_1 < i_2 < ... < i_m \leq n$. The value m is called the length of the subsequence. Such a subsequence is called *ascending* if $A[i_1] \leq A[i_2] < ... < A[i_m]$. For simplicity assume all the values in the array are distinct.

   (a) Design a divide-and-conquer algorithm for solving the longest ascending subsequence problem in time $O(n^2)$. (Hint: Solve a tougher problem. For each position in the array find the cardinality of the longest sequence that ends up with it, and the longest sequence that starts with it. Notice that $n^2 + 2(n/2)^2 + 4(n/4)^2 + \ldots$ is $O(n^2)$.)

   (b) Design a dynamic programming algorithm in time $O(n \log n)$. (Hint: Assume you have a data structure in which *insert(value)* and *find(value)* is $O(\log n)$ operations where the latter operation returns the highest value that was inserted less or equal to $value$.)

3. In parallel computing, to solve a problem, processors communicate with others round by round. In each round, processors can concurrently read or exclusively write(CREW) on shared memory, i.e., multiple processors can read a shared memory cell, but only one can write to a given cell at a time. Given a weighted undirected graph $G = (V, E)$, where $|V| = n$ and edge weight are distinct. There are $|E|$ processors and each processor P_i has the input of a unique edge $E_i = \{u, v\}$ and its weight $w(E_i)$. Every shared memory cell can hold $O(\log n)$ bits. To solve the minimum spanning tree of G, because every processor has only partial information of G, it needs to communicate with others and then decide whether its input edge is in the MST or not. Design an algorithm to solve MST in $O(n \log n)$ *rounds*. Values in the input to the problem are of size $O(\log n)$ bits. A cell can hold at most constant number of such values, i.e., the whole description of G cannot be written in a single cell. The output is each processor determines whether its edge is in the MST or not. (Hint: Notice that in this setting if we have a polynomial of n number of processor each holding a value then the minimum value can be found in $O(\log n)$ time. To use this assumption you have to show how to do it.)

4. After class on Wednesday a student asked me why not use the "obvious" greedy algorithm to solve the Knapsack problem. What will the obvious greedy algorithm be? Give a counter example to show that Knapsack is not amenable to a greedy solution (In Dynamic programming we do not fill the Knapsack incrementally making commitments as we go. We either know the whole

solution or not. We cannot determine a partial solution that we know can be extended.). Show that nevertheless if the Hiker is in a hurry and she fills the Knapsack using the greedy algorithm then the value she carries is greater equal than half of the optimal value.

★ Express your algorithm in a well-structured manner. Use pseudo code and the textbook has good examples to follow. Avoid using a long continuous piece of text to describe your algorithm. Start each problem on a NEW page. Unless specified, you should justify the time complexity of your algorithm and why it works. For grading, we will take into account both the correctness and the clarity. Your answers are supposed to be in a simple and understandable manner and sloppy answers are expected to receiver fewer points.

★ Homework assignments are due on Gradescope. Email attachments or paper submissions are NOT acceptable.

★ Raw photo is NOT acceptable. Upload your homework as a PDF scan by using a scanner or mobile scanner app. Match each problem with your answer on Gradescope. Use dark pen or pencil and your handwriting should be clear and legible.

★ We recommend using LaTeX, LyX or other word processing software for writing the homework. This is **NOT** a requirement but it helps us to grade and give feedback.