ECE 374 B - Fall 2024

Homework 6

- **Submit your solutions electronically on the course Gradescope site as PDF files.** If you plan to typeset your solutions, please use the LaTeX solution template on the course web site. If you must submit scanned handwritten solutions, please use a black pen on blank white paper and a high-quality scanner app (or an actual scanner, not just a phone camera). We will mark difficult to read solutions as incorrect and move on.
- **Every homework problem must be done *individually*.** Each problem needs to be submitted to Gradescope before 6AM of the due data which can be found on the course website: https://ecealgo.com/fa24/homeworks.html.
- For nearly every problem, **we have covered all the requisite knowledge required to complete a homework assignment prior to the "assigned" date.** This means that there is no reason not to begin a homework assignment as soon as it is assigned. Starting a problem the night before it is due a recipe for failure.

Policies to keep in mind

- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use, and you ***must*** write everything yourself in your own words. See the academic integrity policies on the course web site for more details.
- **Being able to clearly and concisely explain your solution is a part of the grade you will receive.** Before submitting a solution ask yourself, if you were reading the solution without having seen it before, would you be able to understand it within two minutes? If not, you need to edit. Images and flow-charts are very useful for concisely explain difficult concepts.

See the course web site (https://ecealgo.com/fa24) for more information.

If you have any questions about these policies,
please don't hesitate to ask in class, in office hours, or on Piazza.

Extra Instructions Solutions to a dynamic programming problem have (at minimum) three things:

- A recurrence relation
- A *brief* description of what your recurrence function represents and what each case represents.
- A *brief* description of the memory element/storage and how it's filled in.

1. A grammar G is a way of generating strings of "terminal" characters from a nonterminal symbol S, by applying simple substitution rules, called productions. If $B \to \beta$ is a production, then we can convert a string of the form $\alpha B \gamma$ into the string $\alpha\beta\gamma$. A grammar is in Chomsky normal form if every production is of the form "$A \to BC$" or "$A \to a$," where A, B, and C are nonterminal characters and a is a terminal character. Design an $O(n^3)$-time dynamic programming algorithm for determining if string $x = x_0 x_1 \dots x_{n-1}$ can be generated from the start symbol S.

   Hint: The solution to this problem is the very famous CYK-algorithm, but the main point of this exercise is to see if you can summarize this algorithm succinctly (must formulate the solution as a recurrence with english description for each case).

2. Suppose you have a circular necklace with n jewels, each with some value, v_i. You wish to sell the jewels individually, but unfortunately, removing jewel i from the necklace breaks the neighboring jewels, making them worthless. Design an efficient algorithm to figure out the maximum revenue you can receive from the circular necklace.

3. We have an $n \times m$ rectangle/array M. Each cell in M has a arbitrary value in the range of $[1, n \cdot m]$. We want to cut M into several pieces of rectangles such that all the grid cells in a single rectangle have the same value. A rectangle can be cut only along one of it's horizontal or vertical grid lines, which will break it into two rectangles. Note that you can only cut one rectangle at a time. And the price of cutting the rectangle is the area of the rectangle (aka the number of grid cells in it). Clearly the worst case is to cut M into $n \times m$ pieces. But we want to find if there is any better solution.

   Describe an algorithm that computes the minimum cost of cutting the input grid M into pieces, following the above rules, so that the if two grid cells is on the same board, they must have the same value. Your algorithm should be as fast as possible.

4. Assume we have an graph G with n vertices and m edges. We have two balls B_1, B_2 that will move along their designed path P_1, P_2. Both balls will start at $P_1[1]$ and $P_2[1]$, and can move only forward along their designed paths. To be more precise, we define a valid move for a ball is either stay in it's current node, or move to the next node on its path. Each ball makes exactly one legal move in each round. A sequence, specifying in each round a valid move for each robot, is a plan.

   The score of a plan is the maximum distance of the two balls from each other at any moment during the execution of the plan.

   Describe an algorithm that computes the minimum score plan for the two balls. Your algorithm should be as fast as possible.