CS3230: Assignment for Week 8

Due: Sunday, 27th Mar 2022, 11:59 pm SGT.

Please upload PDFs containing your solutions (hand-written & scanned, or typed) by 27th Mar, 11:59 pm to `Assignments/Assignment8/Submissions`. Name the file **Assignment8_SID.pdf**, where SID should be replaced by your student ID.

You may discuss the problems with your classmates or read material online, but you should write up your solutions on your own. Please note the names of your collaborators or online sources in your submission; failure to do so would be considered plagiarism.

Note: For dynamic programming algorithms, you are not required to write a pseudocode, but you must write down the recurrence clearly.

1. (1 point) Consider a variant of the knapsack problem with n items where there is an extra parameter $R \in \{1, 2, \ldots, n\}$, and you are supposed to choose at most R items with total weight at most W and as high total value as possible. You want to output the maximum value.

   Explain how you can modify the dynamic programming algorithm for knapsack to solve this problem (in particular, write down the recurrence relating the solution of a problem to the solutions of its subproblems), and analyze the running time of your algorithm.

2. (1 point) You are given an array $A[1\,..\,n]$ of (not necessarily positive) integers. Design and analyze the correctness and running time of a dynamic programming algorithm that computes the largest sum of a subset of elements no two of which are adjacent in the array. (The empty subset is considered to have sum 0.)

3. (1 point) You have n tasks to complete and n helpers. The probability that helper i can complete task j is $p_{i,j} \in [0, 1]$, independently of other helpers and tasks. Each helper can only be assigned to one task (so each task must be assigned to exactly one helper).

   (a) Design and analyze an algorithm running in time $O(n \cdot 2^n)$ that computes the maximum probability of all tasks being completed, where the maximum is taken across all possible assignments.

   (b) What is the running time of a brute-force algorithm that checks all possible assignments and finds one that maximizes the probability of all tasks being completed?