CS3230 Semester 2 2024/2025

Assignment 06
Dynamic Programming

Due: Sunday, 23rd Mar 2025, 11:59 pm SGT.

Instructions:

- Canvas Assignment Submission page: `Assignments/Assignment 6`.
- Please upload PDFs containing your solutions (hand-written & scanned, or typed) by the due date.
- Name the file **Assignment6_SID.pdf**, where SID should be replaced by your student ID.
- You may discuss the problems with your classmates at a high level only. You should write up your solutions on your own (any copying from your co-students or usage of Internet or AI tools is not allowed). Please note the names of your collaborators or any other sources in your submission; failure to do so would be considered plagiarism.
- Question listed as "graded for correctness" (worth 6 points) require complete answers. Other questions (worth 1 point each) will be graded only based on reasonable attempts. However, you should still do these questions, as they are practice questions, which would be useful for exams as well as for your knowledge.
- **Note:** For dynamic programming algorithms, you are not required to write a pseudocode, but you must write down the recurrence clearly.

1. (6 points; graded for correctness) There are n sheets of paper, each with a front side (clearly marked "Front") and a back side (clearly marked "Back"). For $i \in \{1, 2, \dots, n\}$, sheet i has a non-negative integer a_i written on its front side and a non-negative integer b_i written on its back side. Let $A = \sum_{i=1}^{n} a_i$ and $B = \sum_{i=1}^{n} b_i$. A set of sheets $S \subseteq \{1, 2, \dots, n\}$ is called *fantastic* if $\sum_{i \in S} a_i \geq A/2$ and $\sum_{i \in S} b_i \geq B/2$.

   Design and analyze an algorithm that computes the minimum size of a fantastic set. Your algorithm should run in time polynomial in n, A, and B. You must state the running time using the O-notation.

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.) What is the space complexity of your algorithm?

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?