15-451/651 Algorithm Design & Analysis, Spring 2026

Homework #1

Homework #1, Due: Sunday, January 18th at 3:00 pm

Guidelines:

1. This homework has three problems, all graded on completion.
2. In all cases the write-up you submit must be written by you alone. You may not share your written solutions with each other.
3. You can discuss the problem as a group, or consult the internet (in fact, references are provided directly when possible). However, you then must write up your final solutions independently in your own words.
4. When asked to "give an algorithm" or "show a runtime", you should:
   (a) describe your algorithm in English or pseudocode,
   (b) provide a short argument for correctness and running time,
   (c) work with arbitrary, unknown, inputs, instead of on a single instance/example.
5. For all dynamic programming problems, which is all the problems here, you should clearly state the states, transitions, and base cases.
6. Problems 3 has an easier part that the full answer builds off of. It's intended to gives ideas toward getting partial points on quizzes/tests: you can skip the earlier part if you have solutions towards the latter ones, as the DP states are almost entirely the same.

Submission: Submit to `gradescope` using join code that you received by email: please contact us if you did not find this code, or have trouble joining the course.

Your solutions **must** be typeset, **not handwritten**, and submitted as a PDF file. You should use US letter-size paper, a font size no smaller than 10pt, margins no smaller than 1in.

1. (*, actual optimal static binary search tree)

   Recall the optimum binary search tree problem mentioned briefly at the end of Lecture 1: we want to arrange keys 1 to n in a binary tree, while minimizing

   $$\sum_i \textsc{Depth}(i) \cdot w_i$$

   where $\textsc{Depth}(i)$ is the depth of i in the tree, and w_is are given weights. Given an $O(n^3)$ time algorithm for this problem.

   Note that this is not leaf-BST: internal nodes in the binary search tree should also have keys.

   References:

   - `https://en.wikipedia.org/wiki/Optimal_binary_search_tree`
   - `https://chatgpt.com/share/6966e15b-3b60-800c-9312-1ff633d5ad2e`, but up to the $n \leq 300$ part...

2. (**, knapsack with weights from small ranges)

   Consider the value optimization version of knapsack without replacement: given n items with weights $w_1 \dots w_n$ and values $v_1 \dots v_n$, find a subset with weight at most W whose value is maximized. However, instead of W small, all weights are integers in a small range, from x to $x + k$.

   Give an algorithm for solving this problem that runs in $O(n^{10}k^{10})$ time or faster.

   Note:

   - The 10s in exponents are to give some leeway: course staff are aware of $O(n^2k)$.
   - When x is large, e.g. $x > n^{100}$, the $O(nW)$ bound we discussed in class can be much worse than this.
   - the reference below gives $O((n/k)^k)$, which does not meet the runtime we ask here when k is large (e.g. $k = n/10$).

   Reference: `https://atcoder.jp/contests/abc060/tasks/arc073_b`, with 3 replaced by k.

3. (counting counting knapsack)

   We have n items of positive integer weights, and a sum goal S.

   (a) (**, skip if you do part b) Give an algorithm that finds the number of subsets of these n items that sum to S in $O(nS)$ arithmetic operations.

   (b) (***) Compute the **sum** of the answer of part (a) over all subsets of these n elements. That is, we want to sum over all subsets of these n elements, A the number of subsets of A that sum to S. Show that this is still computable in $O(nS)$ arithmetic operations.

   hint: you can do this by adding 2 characters to a solution for part (a).

   Note: the numbers that arise during these computations are enormous. Autograders keep this cost under control by asking for answer mod 998244353. For this course, we ask you to bound the number of arithmetic operations instead: you can assume all arithmetic, in particular, addition, and multiplication, take $O(1)$ time. Division/round are almost never needed for such problems: they are the point where arithmetic cost models can be severely abused.

   Reference: `https://atcoder.jp/contests/abc169/tasks/abc169_f`