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

Homework #6

Homework #6, Due: Sunday, Apr 12 at 9:00pm

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.

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. (*) Show that the weighted majority algorithm works when the outcome space is no longer binary. That is, there is now a space of k outcomes, and you (or an expert) are correct only if the outcome picked is the true one.
2. (**) In this problem we consider caching as an online algorithm. There are n items (numbered $1 \dots n$ for convenience), and you have a cache of size k, with which you can keep k of them. The process then proceeds in rounds (similar to the 'days' in rent-or-by): every turn a number comes up, if it's not in the cache, you need to bring it into the cache at cost 1. Also, if the cache is now full, you need to remove something form cache (could be the item you just brought in).

   Prove that when $k = n-1$, no algorithm can achieve a competitive ratio less than $(n-1)$.
3. (****, Tree packing via MWUs) Recitation 10 ran MWU with shortest path at each step. For this problem, we swap that with MST...

   Show that if an undirected graph G is k-edge-connected (every pair of vertices has k edge-disjoint paths between them), we can find a collection of $k \log n$ spanning trees of G so that no edge of G appears in them more than $100 \log n$ times.

   Note:

   (a) A collection of k edge disjoint trees also serve as a certificate that G is k-edge connected, since there is a s-t path in each tree.
   (b) You can treat k as larger than any polylog factors, e.g. $k \geq n^{0.1}$.
   (c) It's acceptable to have extra factors of $\log n$ in either number of trees, or congestion (max number of times an edge of G appears in the collection in the trees).

   hint: A starting point is showing that in such a graph, for any assignment of positive weights to the edges, the min weight spanning tree has weight at most $10/k$ of the total weight.