# CS 270 Combinatorial Algorithms & Data Structures — Spring 2023

# Problem Set 3

Due: 11:59pm, Friday, February 24th
Solution maximum page limit: 3 pages

See homework policy at `https://cs270.org/spring23/syllabus/#homework-policies`

**Problem 1:** (10 points) Show that the van Emde Boas tree can be modified to support both insertion and predecessor queries in $O(\log\log u)$ expected time, while only using $O(n)$ space. You can use the fact without proof that there is a solution to the dynamic dictionary problem with linear space and $O(1)$ expected update and query time (e.g., using hashing with chaining). **Hint:** use indirection (as described in Lecture 9 when covering y-fast tries).

**Problem 2:** (10 points) Let $w$ be a perfect square. Show that there exist positive integers $m$ and $t$, $m < 2^w$ and $0 \le t \le w$, such that for all $x \in \{0,1\}^{\sqrt{w}}$ we have that

$$\left(\left(\left(\sum_{i=1}^{\sqrt{w}} x_i \cdot 2^{i\cdot\sqrt{w}-1}\cdot\right) \times m\right) \gg t\right) \,\&\, (2^{\sqrt{w}}-1) = \sum_{i=1}^{\sqrt{w}} x_i \cdot 2^{i-1}.$$

That is we can pick $m$ and $t$ so that, if we form a bitvector of length $w$ which has the $\sqrt{w}$ bits of $x$ evenly spread out with a $\sqrt{w}$-spacing of zeroes in between bits, then multiplying by $m$ and bitshifting right by $t$ followed by masking perfectly compresses the bits of $x$ into the rightmost $\sqrt{w}$ bits of a machine word. This provides the proof of a lemma we needed for $O(1)$ time most significant set bit in Lecture 10.

**Problem 3:** (10 points) Give an algorithm for computing the least significant set bit of a given input word $x$ in constant time. You may spend $poly(w)$ time in pre-processing (before seeing $x$) to pre-calculate any special constant values that your algorithm needs.

**Problem 4:** (1 point) How much time did you spend on this problem set? If you can remember the breakdown, please report this per problem. (sum of time spent solving problem and typing up your solution)