CS 224 Advanced Algorithms — Spring 2017

Problem Set 1

Due: 11:59pm, Monday, February 6th

Submit solutions to Canvas, one PDF per problem:
`https://canvas.harvard.edu/courses/21996`

Solution max page limits: One page each for problems 1 and 2, two pages for problem 3

See homework policy at `http://people.seas.harvard.edu/~minilek/cs224/hmwk.html`

Problem 1: (10 points) In class we showed that a van Emde Boas tree in which clusters are stored in hash tables uses $O(n \lg w)$ space, and has $O(\lg w)$ expected query time and update time ("expected" since we are assuming use of a hash table with expected constant query and update time, such as hashing with chaining).

(a) (7 points) Give a family of examples of n items that, when stored in a vEB tree for word size w, consumes space $\Omega(n \lg w)$. In your family of examples, n and w should go to infinity.

(b) (3 points) How would you use indirection to modify vEB trees to solve the static predecessor problem with $O(n)$ space and $O(\lg w)$ expected query time?

Problem 2: (5 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

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

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 in the Lecture 2 notes to compute the most significant bit in constant time.

Problem 3: In the below problems you should assume a word size of $w \ge \lg n$. You should solve these problems using only the data structures and techniques shown in class (in particular, the solution we have in mind *does not* use the dynamic version of fusion trees, which we mentioned in class but did not cover).

(a) (5 points) Give a deterministic sorting algorithm running in time $O(n \lg n / \lg\lg n)$.

(a) (5 points) Give a randomized sorting algorithm running in expected time $O(n\sqrt{\lg n})$. **Hint:** you may use the result from problem 1(b) of this problem set, even if you did not solve it.