# CS 671 (Fall 2020) — Assignment 4

# Due: 11/13/2020

Read Chapters 6, 7.1–7.6, and 12.4 from the textbook.

Since this is the first time we have problems assigned from the textbook, and some of those problems point to papers in the literature where they are solved, I just want to clarify that it is of course *not* appropriate to seek out these references (or any summaries of them) as hints for solutions — this would very much be cheating.

**(1)** Let $\mathcal{R}$ be a range space of VC-dimension $d$. Define $\mathcal{R}' := \{A \cap B \mid A, B \in \mathcal{R}\}$ to be the set of all pairwise intersections of sets in $\mathcal{R}$. Prove that $\mathcal{R}'$ has VC-dimension $O(d)$.

Hint: Use some ideas from our proof of the $\epsilon$-net theorem, including the Sauer-Shelah Lemma. Recall that the lemma states that if $\mathcal{R}$ is a range space of VC-dimension $d$ on a ground set of size $n$, then $|\mathcal{R}| \leq \sum_{k=0}^{d} \binom{n}{k}$. Then apply suitable bounds based on Stirling's approximation. (See, e.g., Appendix B of the textbook.)

If you do not manage to prove an $O(d)$ bound, you will get significant partial credit for proving an $O(d \log d)$ bound.

**(2)** Problem 7.2 from the textbook.

**(3)** Problem 12.24 from the textbook. (Hint: It might help you to first solve the problem for forests instead of general graphs. Among other things, you will get significant partial credit for doing so.) To get anything close to full credit, your $W$ must be polynomial in $n$.

**(4)** As we mentioned in class, the 2-SAT algorithm from Section 6.1 can be naturally generalized to 3-SAT (or $k$-SAT). Since 3-SAT is NP-hard, a polynomial number of steps will not usually suffice. It turns out to be better to randomly restart the algorithm periodically if it hasn't found a solution yet.

Here's the new algorithm:

---

1: **for** $g(n)$ iterations **do**
2: Let **x** be an assignment where each $x_i$ is independently true/false with probability $1/2$.
3: **while** $x$ is not a satisfying assignment, and the number of update steps has been less than $f(n)$ **do**
4: Let $C$ be a uniformly randomly chosen unsatisfied clause.
5: Let $x_i$ be a uniformly random variable in $C$.
6: Update **x** by flipping the value of $x_i$.
7: **if** no solution has been found **then**
8: Output "Probably Unsatisfiable".
9: **else**
10: Output the satisfying solution.

---

Here, we will focus only on 3-SAT, and prove that we can set the parameters such that if the formula is satisfiable, we will find a satisfying assignment in time $O((4/3)^n \text{poly}(n))$. We will derive this in several steps. Notice that even if you don't succeed in proving one part, you can still use it in the next part of the problem.

For the analysis, assume that the formula is satisfiable. As in class, we will consider a suitable (now biased) Markov Chain on the interval of numbers $\{0, \ldots, n\}$.

(a) Derive the probabilities for the resulting Markov Chain.

(b) We will now be more pessimistic, and consider the natural generalization of the Markov Chain to the infinite line $\{0, 1, 2, \ldots\}$.

Let $p(d)$ be the probability that starting at $d$, this new random walk will *ever* reach 0. Using that $p(0) = 1$ and $p(\infty) = 0$, prove that $p(d) = 2^{-d}$ for all $d$.

(c) In the same version as above, prove that the probability of being at 0 after *at most* $3d$ steps, starting from $d$, is at least $q(d) = \frac{1}{2^d \text{poly}(d)}$. (This result says that choosing $f(n) = 3n$ is a safe choice. The reason is that in the pessimistic version, our probability of ever getting to the origin will be at best polynomially higher than our probability of getting there within at most $3n$ steps.) I suggest that you first consider the probability of being at 0 after exactly $3d$ steps, while allowing the walk to visit negative numbers, too.

(d) Prove that the probability that one whole iteration of the outer loop (starting from a random assignment) finds a satisfying assignment in at most $f(n)$ updates is at least $(3/4)^n \cdot \frac{1}{\text{poly}(n)}$.

(e) Determine a suitable function $g(n)$, and prove that with high probability, the resulting algorithm will, in time $O((4/3)^n \cdot \text{poly}(n))$, find a satisfying assignment (when there is one).