Randomized Algorithms 2023
Solutions to Tutorial Sheet 4

1. (a) Our aim is to give a (Las Vegas) randomized algorithm that runs in expected polynomial time and that outputs an assignment that satisfies at least $m(1-2^{-k})$ of the m clauses in the given k-CNF formula, where every clause has exactly k literals.

   Our algorithm simply chooses uniformly at random an assignment to the variables, and repeats this until it finds an assignment that satisfies at least $m(1-2^{-k})$ of the clauses.

   Let X_i, $i = 1, \ldots, m$ denote an indicator random variable, whose value is $X_i = 1$ if the i'the clause is satisfied by a random assigmnent, and $X_i = 0$ otherwise. Let $X = \sum_{i=1}^{m} X_i$ denote a random variable representing the total number of clauses that are satisfied by a u.a.r. random assignment.

   Note that $E[X_i] = (1-2^{-k})$ is the probability that the i'the clause is satisfied, because for each clause there is only one assignment to its k variables (out of 2^k such assignments) that doesn't satisfy that clause. is satisfied by every possible assignment to its variables except one. Hence that the *expected* number of clauses that are satisfied, by linearity of expectation, is:

   $E[X] = \sum_{i=1}^{m} E[X_i] = m(1-2^{-k})$.

   However, this does not yet imply a good bound on the expected number of times we would have to choose a random assignment before one of them satisfies at least $m(1-2^{-k})$ of the clauses.

   To establish this, let us first recall that Markov's inequality tells us that for any non-negative random variable Z, and any $a \geq 1$, $\Pr[Z \geq aE[Z]] \leq \frac{1}{a}$.

   Now, in order for us to appy Markov's inequality to our problem, consider the random variable $Y = m - X$, which counts the total number of clauses that are *not* satisfied by a random assignment. Note that Y is a non-negative random variable, and note that an easy (symmetric) argument shows that $E[Y] = m2^{-k}$.

   Now, Let $a = 1 + \frac{1}{m+1}$. By Markov's inequality, it follows that $\Pr[Y > aE[Y]] = \Pr[Y > E[Y] + \frac{1}{m+1}E[Y]] = \Pr[Y > m2^{-k} + \frac{m}{m+1}2^{-k}] \leq \frac{1}{a} = \frac{m+1}{m+2}$.

   However, now observe that Y is actually an *integer* random variable, and hence since $E[Y] = m2^{-k}$, and since $\frac{m}{m+1}2^{-k}$ is strictly less than a multiple of 2^{-k}, we must have $\lceil m2^{-k} \rceil = \lceil m2^{-k} + \frac{m}{m+1}2^{-k} \rceil$. Hence, we have that $\Pr[Y > E[Y]] \leq \frac{m+1}{m+2}$. Hence, we have that $\Pr[m - Y < m - E[Y]] \leq \frac{m+1}{m+2}$. Hence, since $X = m - Y$, and since by linearity $E[X] = m - E[Y]$, we have that $\Pr[X < E[X]] \leq \frac{m+1}{m+2}$.

   Hence $\Pr[X \geq E[X]] \geq \frac{1}{m+2}$. But this means that for each u.a.r. random assignment, the probability that it satisfies at least $E[X] = m(1-2^{-k})$ clauses is at least $\frac{1}{m+2}$. Hence, the expected number of random trials we would need before a random assignment succeeds in satisfying $m(1-2^{-k})$ clauses behaves like a geometric random variable with success probability $p = \frac{1}{m+2}$ in each trial. As we know, the expectation of such a geometric random variable is $\frac{1}{p} = m + 2$.

Hence, the expected number of random assigments we would have to try before we find one that satisfies at least $m(1-2^{-k})$ clauses is $m+2$.

This therefore yields a randomized (Las Vegas) algorithm for computing such an assigment, whose expected running time is polynomial in the encoding size of the input k-CNF formula, because we need, in expectation $m+2$ assignments, and each assigment requires sample n u.a.r. random bits (truth assigments), for the n boolean variables of the k-CNF formula.

(b) Derandomizing this algorithm using the method of conditional expectations follows a very similar approach as we took in lectures for derandomizing the algorithm for computing a cut in a graph that cuts at least $1/2$ the edges.

Specifically, suppose we are given a k-CNF formula φ, with m clauses, over n boolean variables $x_1, \ldots, x_n$.

We will sequentially assign truth values to x_1, then x_2, then x_3, and so on, such that in each stage we will make sure that we are not decreasing the conditional expectation of the number clauses that are satisfied by assignment to the remaing variables.

Let X_φ be the random variable that denotes the expected number of clauses in φ that are satisfied by a u.a.r. random assigment to all variables.

Let us consider the conditional expectation of X_φ, conditioned on a particular truth assignment, $b \in \{0,1\}$, to variable x_1. $E[X_\varphi \mid x_1 = b]$ denote this conditional expectation.

However, let φ_b denote the new CNF formula obtained from φ by assigning the truth value $x_1 = b$, to all occurences of the variable x_1 in φ. This can make some clauses true, namely those where x_1 appears with the same sign as b, and it can reduce the size of some other clauses where x_1 appears but with a different sign than b. Hence φ_b is really just a different "reduced" CNF formula. Moreover, it is clear that $E[X_\varphi \mid x_1 = b] = E[X_{\varphi_b}]$.

Note that we can easily calculate $E[X_{\varphi_b}]$, for each $b \in \{0,1\}$, because for each clause in φ_b (where each clause may contain k *or fewer* literals), we can calculate the probability that a u.a.r. random assignment satisfies that clause, and using linearity of expectation we can sum all of these probabilities to obtain the expected number of clauses of φ_b that are satisfied by a u.a.r random assigment.

Now, clearly, $E[X_\varphi] = \frac{E[X_{\varphi_1}]}{2} + \frac{E[\varphi_0]}{2}$

Thus, one of $E[X_{\varphi_1}]$ and $E[X_{\varphi_0}]$ must be at least as large as $E[X_\varphi]$.

We can hence choose the truth value of x_1 to be the truth value that yields the larger of these two conditional expectations, i.e., $x_1 = \arg\max_b E[X_{\varphi_b}]$.

Having chosen the value of x_1 this way, let us suppose that value is b_1. We can next continue to assign a truth value $b_2 \in \{0,1\}$ to x_2, so as to maximize $E[X_{\varphi_{b_1}} \mid x_2 = b_2]$. But again, fixing the value of x_2 to b_2 yields a new CNF formula φ_{b_1,b_2}, which is a "subformula" of φ_{b_1}, and clearly $E[X_{\varphi_{b_1}} \mid x_2 = b_2] = E[X_{\varphi_{b_1,b_2}}]$. Again, we can easily calculate $E[X_{\varphi_{b_1,b_2}}]$, so we can fine which value of b_2 maximizes this.

In this way, we can continue to assign values $b_1, b_2, b_3, \ldots$, sequentially to each variable $x_1, x_2, x_3, \ldots$, in each case making sure that the conditional expectation of the number of clauses that are satisfied is not decreased.

At the end this results in an assigment $(b_1, \ldots, b_n)$ to all n variables which must therefore satisfy at least $E[X_\varphi] = m(1 - 2^{-k})$ clauses.

2. (a) To see that $S(\sigma)$ must be an independent set, note that if $i, j \in S(\sigma)$ then *both* vertex i and vertex j appear in the permutation σ before any of their neighbors. But this means that i and j themselves cannot be neighbors of each other, because one of them appears after the other in σ.

   (b) One possible randomized algorithm to generate a permutation σ u.a.r. would be to first choose the first entry i_1 in the permutation, u.a.r., from the entire set $\{1, \ldots, n\}$ of vertices. Next, we choose the next element i_2 u.a.r. from the remaining set $\{1, \ldots, n\} \backslash \{i_1\}$, and so on, until we have generated the entire permutation $i_1, i_2, i_3, \ldots, i_n$.

   (This is not the most efficient way to generate such a permutation u.a.r., but it is efficient enough for our purposes, and in particular it requires at most a polynomial number of coin flips as a function of n.)

   Now we show that the expected size of $S(\sigma)$ is $\sum_{i=1}^{n} \frac{1}{d_i+1}$. To do this, let us define an indicator random variable X_i for each node $i \in \{1, \ldots, n\}$, where $X_i = 1$ if node i appears in the random permutation σ prior to any of its neigbors in the graph G, and otherwise $X_i = 0$.

   Note that $X = \sum_{i=1}^{n} X_i$ is a random variable denoting the total number of nodes that appear before all their neighbors in a u.a.r. random permutation σ. In other words $X = |S(\sigma)|$.

   Note, crucially, that the probability that node i appears in a u.a.r. random permutation σ prior to any of its d_i neighbors is precisely $\frac{1}{d_i+1}$.

   Hence $E[X_i] = \frac{1}{d_i+1}$. Hence, by linearity of expectation $E[X] = \sum_{i=1}^{n} \frac{1}{d_i+1}$.

   (c) We have established that $E[X] = \sum_{i=1}^{n} \frac{1}{d_i+1}$.

   Hence, we can simply apply the probabilistic method to conclude that there must exist an independent set of this size. This is because we know, for any random variable X with finite expectation that $\Pr[X \geq E[X]] > 0$ (see slides for lecture 11, or Lemma 6.2 in the textbook).

3. (a) This observation follows easily because:

$$\begin{aligned} e \cdot p(d+1) \leq 4dp &\Leftrightarrow e(d+1) \leq 4d \\ &\Leftrightarrow \frac{e}{4} \leq \frac{d}{d+1} \\ &\Leftrightarrow d \geq 3 \quad \text{(because } d \text{ is an integer)} \end{aligned}$$

   Hence, for any $d \geq 3$, $e \cdot p(d+1) \leq 4dp$. Hence the condition $e \cdot p(d+1) \leq 1$ is less stringent than the condition $4dp \leq 1$.

(b) Now let $x_i = \frac{1}{d+1}$, for all $i = 1, \ldots, n$. Observe that the general Lovasz Local Lemma (Theorem 6.17 in the textbook) then implies that, under the condition that for all i, $\Pr[E_i] \leq \frac{1}{d+1} \prod_{(i,j)\in E}(1 - \frac{1}{d+1})$, we have

$$\Pr(\bigcap_{i=1}^{n} \overline{E}_i) \geq \prod_{i=1}^{n}(1 - \frac{1}{d+1}) > 0. \tag{1}$$

But note that $\prod_{(i,j)\in E}(1 - \frac{1}{d+1}) \geq (1 - \frac{1}{d+1})^d \geq \frac{1}{e}$.

Hence, in order to conclude (1), it suffices to assume that $\Pr[E_i] \leq \frac{1}{e(d+1)}$ for all i. In other words, it suffices to assume $\Pr[E_i] \leq p$, for all i, where $p \leq \frac{1}{e(d+1)}$, or equivalently where $ep(d+1) \leq 1$.