1 MaxSat

a)

We will use the theorem 6.4 which declares that *"Given a set of m clauses, let k_i be the number literals in the i^{th} clause for $i = 1, ..., m$. Let $k = min_{i=1}^{m} k_i$. Then there is a truth assignment that satisfies at least $\sum_{i=1}^{m}(1 - 2^{-k_i} \leq m(1 - 2^{-k}))$ clauses"*. Based on that, we declare the following Las-Vegas algorithm:

Algorithm 1 Las-Vegas SAT algorithm

```
Require: k formula with m clauses
  output = None
  while output is None do
      Assign True(1) or False(0) to each variable of the m clauses independently and uniformly at random.
      if Assignment satisfies at least $m(1 - 2^k)$ clauses then
          output = Assignment
      end if
  end while
  return output
```

At this point we proving the theorem 6.4 just for defining the necessary random variables we will need. We define S_i the indicator random variable that the i^{th} clause satisfies based on the assignment. The probability of satisfying the assignment is one of all the possible 2^k different assignments we can make with k literals. So $E[S_i] = Pr(S_i = 1) = 1 - 2^{-k}$. Also, we define the sum of all possible clauses that satisfy their assignments as $S = \sum_{i=1}^{m} S_i$. So by linearity of expectations, the expected number of clauses that satisfy their assignments is:

$$E[S] = E[\sum_{i=1}^{m} S_i] = \sum_{i=1}^{m} E[S_i] = \sum_{i=1}^{m}(1 - 2^{-k}) = m \cdot (1 - 2^{-k})$$

Our proposed Las-Vegas algorithm repeats a number of iterations and in every iteration it calculates m clauses with k literals which costs $O(mk)$. Therefore, we need to find out the expected number of iterations. As we noticed, the repetition of the algorithm until we find at least $m \cdot (1 - 2^{-k})$ clauses that satisfy their assignments with a probability p is by definition geometric with p as parameter. So the expected number of iterations is $\frac{1}{p}$ and we can estimate the probability based on the expected number of clauses as follows:

$$E[S] = m \cdot (1 - 2^{-k}) = \sum_{i=1}^{m \cdot (1-2^{-k})-1} (1 - p) + \sum_{i=m\cdot(1-2^{-k})}^{m} p \Longrightarrow$$
$$m \cdot (1 - 2^{-k}) = (1 - p) \cdot [m \cdot (1 - 2^{-k}) - 1] + p \cdot [m - m \cdot (1 - 2^{-k})] \Longrightarrow$$
$$1 = p \cdot (1 - m + 2^{1-k} \cdot m) \Longrightarrow$$
$$1 \leq p \cdot 2^{-k} \cdot m \Longrightarrow p \geq \frac{2^k}{m}$$

So the total expected running time is calculated as $O(mk) \cdot \frac{1}{p} = O(mk) \cdot \frac{m}{2^k} = O(\frac{m^2 k}{2^k})$

b)

We will derandomize the algorithm of subquestion (a) using the conditional expectations as described in chapter 6.3 of the book with the same notation as in the previous subquestion. We also define as lt_i the random variable of the value of the literal i in a clause. The goal is to build a deterministic algorithm to output an assignment which satisfies at least $m(1-2^k)$ clauses. Based on that, we declare the following deterministic algorithm: Based on the lecture 22 we have to prove through induction that the output of our algorithm satisfies clauses at least equal with $E[S]$. Hence, it should be proven that:

$$E[S|lt_1 = l_1, ..., lt_a = l_a] \geq E[S] \tag{1}$$

where a is the induction variable.

Algorithm 2 Deterministic SAT algorithm

```
Require: k formula with m clauses
  output = None
  for $i = 1$ to $k$ do
     We have already assigned values for all $i-1$ literals in each clause. $lt_1 = l_1, lt_2 = l_2, ..., lt_{i-1} = l_{i-1}$
     /* Calculate $E[S|lt_1 = l_1, lt_2 = l_2, ..., lt_{i-1} = l_{i-1}, lt_i = l]$ */
     for $l = 0$ to 1 do
        $l_i = l$
        Replace all literals $lt_1 = l_1, lt_2 = l_2, ..., lt_{i-1} = l_{i-1}, lt_i = l_i$ of the formula and based on the satisfied clauses defined as $s$, sum them up and return the result $r_i = \sum_s (1 - \frac{1}{2^s})$
     end for
     if $r_0 > r_1$ then
        $lt_i = 0$
     else
        $lt_i = 1$
     end if
  end for
  return $lt_1, lt_2, ..., lt_k$
```

Base case We choose a = 1, so by the law of total expectation we have:

$$E[S] = \frac{1}{2} \cdot E[S|lt_1 = 0] + \frac{1}{2} \cdot E[S|lt_1 = 1]$$

which means that for one of the two choice we made for lt_1 ($l_1 = 0$ or $l_1 = 1$) we get $E[S|lt_1 = l_1] \geq E[S]$
Induction case We hypothesize that $E[S|lt_1 = l_1, ..., lt_{a-1} = l_{a-1}] \geq E[S]$ and by he law of total expectation we have:

$$E[S|lt_1 = l_1, ..., lt_{a-1} = l_{a-1}] = \frac{1}{2} \cdot E[S|lt_1 = l_1, ..., lt_a = 0] + \frac{1}{2} \cdot E[S|lt_1 = l_1, ..., lt_a = 1]$$

which means that for one of the two choice we made for lt_a ($l_a = 0$ or $l_a = 1$) we get $E[S|l_1, ..., lt_a] \geq E[S]$ Therefore, based on this induction we have shown that equation (1) is true and through the subquestion of (a) we get that the output of our algorithm satisfies clauses at least equal with $E[S] = m \cdot (1 - 2^k)$.

As we did in subquestion (a) the proposed deterministic algorithm repeats a number of iterations and in every iteration it needs m clauses with k literals for the calculation of $E[S|lt_1 = l_1, ..., lt_i = l]$ which costs $O(mk)$. The number of iterations is set as $i \in [k]$ so the final expected running time is $O(mk \cdot k) = O(mk^2)$.

2 Tournaments

a)

We define as $G_T(V,E)$ a directed graph representing the tournament where $V \in [n]$ is the number of total nodes and $E \in [m]$ is the number of total edges. Also, we define as R a uniformly at random ordering of n players called ranking. Moreover, we define the indicator random variable D_i of the event that the i^{th} edge of tournament G_T disagrees with the i^{th} edge of ranking R. Based on that, we define as D the sum of all the possible disagreements between tournament and ranking, so $D = \sum_{i=1}^{m} D_i$. The expected value of $E[D_i] = Pr(D_i = 1) = \frac{1}{2}$ since there are two possible options in the sample space. Either the two graphs agree or disagree. Using the linearity of expectations:

$$E[D] = E[\sum_{i=1}^{m} D_i] = \sum_{i=1}^{m} E[D_i] = \sum_{i=1}^{m} \frac{1}{2} = \frac{m}{2}$$

In conclusion, a ranking R exists that disagrees with at most 50% of the edges.

b)

We use the definitions from the sub question (a). For the tournament $G_T(V,E)$ we know that the number of edges is $m = \binom{n}{2}$ so $E[D] = \frac{m}{2} = \frac{\binom{n}{2}}{2}$. Using the Chernoff bound, we bound the probability that a tournament's edges disagree with the ones of a random ranking by at least 49% as follows:

$$\begin{aligned} Pr(D \le \frac{49}{100}\cdot m) = Pr(D \le \frac{49}{100}\cdot\binom{n}{2}) &= Pr(D \le (1-0.02)\cdot\frac{\binom{n}{2}}{2}) \Longrightarrow \\ &= Pr(D \le (1-0.02)\cdot E[D]) \Longrightarrow \\ &\le e^{-\frac{E[D]\cdot(0.02)^2}{2}} \le e^{-n^2} \end{aligned}$$

However, we need to prove that there is a valid probability for all possible permutations of rankings $(n!)$ that disagrees with $G_T(V,E)$ so with the help of union bound it should be proven that $n!\cdot e^{-n^2} < 1$. Hence,

$$\begin{aligned} n!\cdot e^{-n^2} &< 1 \xrightarrow{ln(\cdot)\nearrow} \\ ln(n!\cdot e^{-n^2}) &< 0 \Longrightarrow \\ ln(n!) - n^2 &< 0 \xrightarrow{ln(n!)\approx nln(n)-n} \\ nln(n) - n &< n^2 \qquad \textit{True, for sufficient large n} \end{aligned}$$

In conclusion, we prove that for sufficiently large n, there exists a tournament such that every ranking disagrees with at least 49% of the edges in the tournament.

3 Sample and Modify

a)

As we know, in the first stage of "sample and modify" technique, we construct a random structure that does not have the required properties. In our case, based on the hint we construct a complete graph of n nodes K_n coloring each edge independently and uniformly choosing the color from a palette of 3 colors.

In the second stage, we modify the random structure so that it does have the required property. In our case the required property is 3-coloring the complete graph K_n in a way that a subgraph of it K_m has no monochromatic k complete graphs. In order to do that we will remove randomly nodes from the k complete graphs that exists in the K_n we produced in the previous step. By doing that we remove the K_k from the original K_n graph. To prove that, we have to show that the expected number of edges m left after the removal will be the equal to $n - E[M]$, where we define as M the random variable of the number of monochromatic k complete graphs $(n > k)$. Based on lecture 22, the number of total k complete graphs is $\binom{n}{k}$, while the number of edges in each complete graph is $\binom{k}{2}$. The sample space of colors all edges of a k complete graph can take is $3^{\binom{k}{2}-1}$. Hence, the probability of getting one edge with the same coloring as all the others is $p = \frac{1}{3^{\binom{k}{2}-1}}$. So the expected value of M, $E[M]$ can be calculated as following:

$$E[M] = \binom{n}{k} \cdot p = \binom{n}{k} \cdot \frac{1}{3^{\binom{k}{2}-1}}$$

Given that $m = n - \binom{n}{k} \cdot \frac{1}{3^{\binom{k}{2}-1}}$ we have:

$$m = n - \binom{n}{k} \cdot \frac{1}{3^{\binom{k}{2}-1}} = n - E[M]$$

So we prove that is possible to 3-color the complete graph K_n in a way that a subgraph of it K_m has no monochromatic k complete graphs.

b)

As we know, in the first stage of ”sample and modify” technique, we construct a random structure that does not have the required properties. In our case, based on the hint we sample independently nodes with probability $p = \frac{ln(d+1)}{d+1}$ and put them in set S.

In the second stage, we modify the random structure so that it does have the required property. In our case the required property is to put those nodes v which neither them nor their neighbors were part of set S in the first part. We define as A the random variable of the total number of nodes chosen in the first step as part of the dominating set S. Those nodes follows a Binomial distribution with expected value of $E[A] = n \cdot p = n \cdot \frac{ln(d+1)}{d+1}$. Moreover, we define the random value of the total number of nodes chosen in the second step as B. The total number of those nodes are all nodes v which were not chosen in first step and their neighbors. The probability to get one such node is bounded as follows:

$$\prod_{i=1}^{d+1}(1-p) = (1-p)^{d+1} \xrightarrow{(1-x)\leq e^{-x}} e^{-p\cdot(d+1)} = \frac{1}{d+1}$$

Hence the expected value of nodes chosen in the second step is $E[B] \leq \frac{n}{d+1}$. So, the total expected value of the nodes in the dominating set S is by linearity of expectations:

$$E[A+B] = E[A] + E[B] \leq n \cdot \frac{ln(d+1)}{d+1} + \frac{n}{d+1} = \frac{n(ln(d+1)+1)}{d+1}$$

4 Threshold for triangles

a)

Based on the exercise,we define a graph $G_{n,p}$ with n vertices and each pair of vertices independently connected by an edge with probability p. Also, we define $t_1, t_2, ..., t_{\binom{n}{3}}$ be an enumeration of all triplets of vertices in the graph. Moreover, we define X_i as the indicator random variable such that $X_i = 1$ event corresponds to if the the three edges if the triplet t_i appear in the graph, so that t_i forms a triangle. Last, we define X as the random variable indicating all the possible triangles in the graph $X = \sum_{i=1}^{\binom{n}{3}} X_i$.

Using the linearity of expectations for X random variable and due to the fact that picking three edges from the $G_{n,p}$ to form a triplet t_i is given by the probability $Pr(X_i = 1) = p^3$ we imply that:

$$E[X] = E[\sum_{i=1}^{\binom{n}{3}} X_i] = \sum_{i=1}^{\binom{n}{3}} E[X_i] = \sum_{i=1}^{\binom{n}{3}} Pr(X_i = 1) = \sum_{i=1}^{\binom{n}{3}} p^3 = \binom{n}{3} \cdot p^3$$

b)

Using the subquestion (a) and the union bound we have:

$$Pr(X > 0) = Pr[\bigcup_{i=1}^{\binom{n}{3}} Pr(X_i = i)] \leq \sum_{i=1}^{\binom{n}{3}} Pr(X_i = 1) = \sum_{i=1}^{\binom{n}{3}} E[X_i] \xrightarrow{(a)}$$

$$= E[X] = \binom{n}{3} \cdot p^3 = \frac{n(n-1)(n-2)}{3!} \cdot p^3 = O((np)^3)$$

In conclusion, if np goes to 0 then since $Pr(X > 0) = O((np)^3)$, $Pr(X > 0)$ will go to 0 respectively.

c)

Based on the fact that X_i is an indicator random variable we know that $E[X_i] = E[X_i^2]$. Hence, using subquestion (a) and $0 < p < 1$ we have that:

$$Var[X_i] = E[X_i^2] - E[X_i]^2 = p^3 - (p^3)^2 = p^3 - p^6 = p^3(1 - p^3) \leq p^3$$

d)

Having 2 triplets of nodes, there are 3 different ways to interact. Firstly, they might totally overlap sharing all 3 nodes which is trivial and not interesting. Secondly, they might overlap 1 out of 3 nodes sharing no edge together so their respective random variable X_i and X_j are independent and as a result $Cov(X_i, X_j) = 0$. Thirdly, they might overlap 2 out of their 3 nodes sharing 1 edge together. If they overlap 2 nodes, that implies we need 4 distinct nodes to build these 2 triples and the total number to choose 4 distinct nodes out of n choices is $\binom{n}{4} = O(n^4)$. Hence, their co variance metric $Cov(X_i, X_j)$ takes the following form:

$$\begin{aligned}
Cov(X_i, X_j) &= E[X_i \cdot X_j] - E[X_i] \cdot E[X_j] \xrightarrow{E[X_i]=E[X_j]=p^3} \\
&= E[X_i \cdot X_j] - E[X_i]^2 \Longrightarrow \\
&= Pr(X_i = 1 \cap X_j = 1) - E[X_i]^2 \Longrightarrow \\
&= Pr(X_i = 1 | X_j = 1) Pr(X_j = 1) - E[X_i]^2 \Longrightarrow \\
&= p^2 \cdot p^3 - (p^3)^2 = p^5 - p^6
\end{aligned}$$

e)

Using both subquestion (b) and (d) and the definition of variance we have that:

$$\begin{aligned}
Var[X] = Var[\sum_{i=1}^{\binom{n}{3}} X_i] &= \sum_{i=1}^{\binom{n}{3}} Var[X_i] + \sum_{i=j=1, i\neq j}^{\binom{n}{3}} Cov(X_i, X_j) \Longrightarrow \\
&= \binom{n}{3} \cdot O(p^3) + O(n^4) \cdot O(p^5 - p^6) \Longrightarrow \\
&= O(n^3) \cdot O(p^3) + O(n^4) \cdot O(p^5 - p^6) \Longrightarrow O(n^3 \cdot p^3 + n^4 \cdot (p^5 - p^6))
\end{aligned}$$

f)

Using both subquestion (b) and (d) and given that X is a non-negative integer-value random variable from Theorem 6.7 we have that:

$$\begin{aligned}
Pr(X = 0) &= \frac{Var[X]}{E[X]^2} \xrightarrow{(b),(d)} \\
&= \frac{O(n^3 \cdot p^3 + n^4 \cdot (p^5 - p^6))}{O(np^3)^2} \Longrightarrow \\
&= O(\frac{n^3 \cdot p^3 + n^4 \cdot (p^5 - p^6)}{(np)^6}) \Longrightarrow \\
&= O(\frac{1}{(np)^3} + \frac{1-p}{n \cdot np})
\end{aligned}$$

In conclusion, if np goes to 0 then, $Pr(X > 0)$ will go to ∞ respectively.