Exercise 1

a)

The model for the ILP is the following:

$$\begin{aligned}
\text{minimize} \quad & M \\
\text{s.t.} \quad & \sum_{e \in E} x_{e,h} = 1 \quad \forall\ h \in H \\
& M \geq \sum_{h \in H_e} x_{e,h} \forall\ e \in E \\
& x_{e,h} \in \{0,1\} \quad \forall\ (e,h) \in E \times H : h \in H_e \\
& M \in \mathbb{N}
\end{aligned}$$

where $x_{e,h}$ is 1 if the elf e is assigned to the house h. To relax the model from ILP to LP, we just relax the last two constraints, i.e. $x_{e,h} \in [0,1], \quad M \in \mathbb{R}$.

b)

We interpret the variables $x_{e,h}$ in the LP relaxation as probabilities; we obtain the solution by setting variables to 1 independently with probability equal to the fractional value. In expectation, our algorithm is as good as the LP optimum: $\mathbb{E}[alg] = \mathbb{E}[M] = \sum_{h \in H_e} \Pr(e,h) = \sum_{h \in H_e} x_{e,h} = OPT_{LP}$, given e the busiest elf. In this situation, the probability that a house h is not matched with an elf is $\Pr[\textit{house } h \textit{ has no elves}] = (1 - x_{e_1,h}) \times ... \times (1 - x_{e_m,h}) \leq (1 - \frac{1}{k})^k \leq \frac{1}{e}$ where k is the number of elves that can deliver at house h. Since our algorithm has to return a feasible solution, we repeat the process $\ln(4n)$ times (with $n = |H|$) to ensure (with high probability) that no house is left without an elf. After $\ln(4n)$ repetitions the probability that a house h is not matched with any elf is $\Pr[\textit{house } h \textit{ has no elves}] \leq (\frac{1}{e})^{\ln(4n)} = \frac{1}{4n}$. The probability of having a house $\tilde{h}$ with no elf after $\ln(4n)$ repetitions is $\Pr(\exists\ \tilde{h}\ \textit{without elf}) \leq n \times \frac{1}{4n} = \frac{1}{4}$. This means that, after $\ln(4n)$ repetitions, we have a probability equal to 3/4 of having all the houses matched with an elf. Thus, on expectation, we have to run this algorithm twice to have all the houses matched with an elf.

c)

To prove that our solution is unlikely to deliver bad results, we prove that with high probability (at least $1 - \frac{1}{m}$) the resulting solution is no worse than $O(\ln m + OPT_{LP})$, i.e. we have to prove that for some constants $a, b \in \mathbb{R}^+$, $\Pr[M \geq bOPT_{LP} + a\ln(m)] \leq \frac{1}{m}$. In point b we proved that $OPT_{LP} = \mathbb{E}[M] = \mu$ and thus we have that $\Pr[M \geq bOPT_{LP} + a\ln(m)] = \Pr[M \geq b\mu + a\ln(m)] = \Pr[M \geq (1 + (b + \frac{a\ln(m)}{\mu} - 1))\mu] \leq exp(\frac{-\mu\ min\{\epsilon,\epsilon^2\}}{3})$, where $\epsilon = (b + \frac{a\ln(m)}{\mu} - 1)$. If we choose $b \geq 2$ we have that $(b + \frac{a\ln(m)}{\mu} - 1) = \epsilon \geq 1$, thus $min\{\epsilon, \epsilon^2\} = \epsilon$. Finally we can say that $\Pr[M \geq (1+\epsilon)\mu] \leq exp(\frac{-\mu\ \epsilon}{3}) = exp(\frac{-\mu}{3}(b + \frac{a\ln(m)}{\mu} - 1)) = exp(-(\frac{(b-1)\mu}{3} + \frac{a\ln(m)}{3}))$. Since we have $b \geq 2$ (to have $\epsilon \geq 1$) we can choose $a \geq 3$ to have $(\frac{(b-1)\mu}{3} + \frac{a\ln(m)}{3}) \geq \ln(m) \implies exp(-(\frac{(b-1)\mu}{3} + \frac{a\ln(m)}{3})) \leq \frac{1}{m}$. So we have that $\Pr[M < 2OPT_{LP} + 3\ln(m)] \geq 1 - \frac{1}{m}$.

Exercise 2

To prove that the problem given to us by Santa Claus is very difficult to solve in reasonable time, we make a reduction from a known difficult problem, 3-SAT. If we are able to make a poly-time reduction from 3-SAT to Santa's problem and if we are able to solve Santa's problem in polynomial time, then we can solve 3-SAT in polynomial time, which is impossible unless $P = NP$.

Given an instance of a 3-SAT problem we create an instance of the Santa's problem as follow: the set P of the pieces used to build the sleighs is composed of $n + 187$ items where n is the number of variables in the 3-SAT formula. We map the *i-th* variable of the 3-SAT problem to the piece $p_i \in P$ and we add other 187 pieces (187 is the number of different sleighs that we need to build). For instance if the 3-SAT problem has 3 variables we take 190 pieces, that could be: a front light, a seat, a window, and 187 different types of back lights; when creating the instance of the Santa's problem we map variable x_1 to the front light, variable x_2 to the seat and variable x_3 to the window. Now we represent the *i-th* clause of the 3-SAT as a constraint of the Santa's problem by adding the constraint: "Any decent sleigh has a *piece mapped to* x_l or a *piece mapped to* x_m or a *piece mapped to* x_n" (assuming that clause i has variables x_l, x_m and x_n). The input of the Santa's problem will be the set of constrains created as mentioned. The output of the Santa's problem is going to be the set of items that are good to build each sleigh. We take one of the sleighs (the *j-th* sleigh) and we build the solution of the 3-SAT problem as follow: for every i, if the *i-th* piece is present on the selected sleigh, we put the associated x_i variable to 1, otherwise we set it to 0. We can ignore the variables associated with the 187 added pieces (the back lights in the previous example, we will explain soon why we added them) as they were not part of the 3-SAT problem. Since the *j-th* sleigh satisfies all the constraint (as every other sleigh does), the set of variables x_i satisfy the 3-SAT formula.

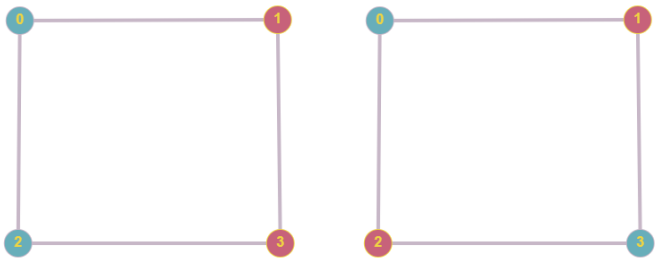
The 187 different type of pieces are needed in order to guarantee that each sleigh will be different: for example, assuming that the 3-SAT problem has only one solution (or, in general, less than 187 solutions), the algorithm that solves the Santa's problem will not be able to build 187 sleighs with just n types of pieces (if the solution for the 3-SAT problem is only one, there is only one set of pieces that satisfies all the constraints, thus we cannot build 187 different sleighs). This problem will lead the algorithm that solves Santa's problem to output "no" (because he cannot build 187 different sleighs), but we know by assumption that at least one solution exists. To solve this problem we add 187 different types of pieces besides the n coming from the variables; this ensures us that even if only one solution exists, the algorithm can build 187 different sleighs by taking the existing solution and adding to it one of the 187 (different) pieces. Note that the 187 added pieces do not appear in any constraint, thus they do not interfere in any way with the satisfiability, they only let the algorithm create different sleighs if it can build at least one, but not 187.

For instance, we may have the following 3-SAT formula: $(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_5 \vee \neg x_6)$, which has many solutions, among which there is (1,0,1,0,0,0). We consider an instance of the Santa's problem with 193 different pieces (a front light,a seat, a door, a window, a wheel, a roof and 187 different types of back lights); the input constraints will be: "Any decent sleigh has a front light, or a seat, or it does not have a door", "Any decent sleigh has a front light, or a door, or it does not have a window", "Any decent sleigh does not have a door, or it does not have a wheel, or it does not have a roof". The algorithm that solves Santa's problem will output the set of pieces needed to build every sleigh and we consider one of them (for instance the first one), that might be composed by a front light, a door and the back light of type 10 (this set of pieces satisfies all the constraints). A solution for the 3-SAT is indeed (1,0,1,0,0,0).

Since we can use the algorithm that solves Santa's problem to solve 3-SAT using a poly-time reduction, no poly-time algorithm can exist to solve Santa's problem unless $P = NP$.

Exercise 3

1)



(a) *Graph in equilibrium case* (b) *Graph in optimal case*

Figure 1: Colors represent the two possible choices for the players, LEFT and RIGHT.

The cut game we designed is composed by four players (nodes 0,1,2,3) linked by edges with unit weights to form a square. The two possible choices (LEFT, RIGHT) are represented by the colors of the nodes. In the figure 1a we have an equilibrium (it is one of the equilibrium with the lowest payoff, i.e. an equilibrium in which the payoff for each player is half of the maximum, which, as we will prove in section 3.2, is the lowest payoff we can obtain in a pure Nash equilibrium) and the payoff for each player is 1, while in the optimal case (figure 1b), the payoff is 2 for each player. Thus, the Price of Anarchy (PoA) is 2.

2)

In a pure Nash equilibrium $U(v_i) \geq \frac{1}{2} \sum_{e \in N_i} w_e$. We can prove this by contradiction, in fact, if we are in a pure Nash equilibrium and $U(v_i) < \frac{1}{2} \sum_{e \in N_i} w_e$, v_i can improve its payoff by changing side, but that means that we were not in a pure Nash Equilibrium. In the worst case we are in an equilibrium such that, $\forall v_i,\ U(v_i) = \frac{1}{2} \sum_{e \in N_i} w_e$, i.e. half of the weight is "directed" on the same side as v_i, while the best is to have, for every player, all of the neighbours on the other side, i.e. $\forall v_i,\ U(v_i) = \sum_{e \in N_i} w_e$. This means that the PoA of cut games, in the general case, is:

$$PoA \leq \frac{\sum_{v_i \in V} \sum_{e \in N_i} w_e}{\sum_{v_i \in V} \frac{1}{2} \sum_{e \in N_i} w_e} = 2$$

Exercise 4

The given problem is similar to the center selection problem seen during the lectures, but this time we have fixed locations for the antennae.

1)

Since we solved point 3, we skipped this part.

2)

To find a 3-approximation algorithm, we extend the greedy center selection algorithm seen during the lectures; at first we apply the greedy center selection algorithm to the set of cities which will give us a set of k *best cities* among them, then we select the closest antenna for each of the k selected city.

We prove that this is a 3-approximation algorithm by observing that the distance from any city c_i to its closest *best city* bc_i (found during the first phase of the algorithm) is at most 2 OPT (where OPT is the optimal value). This can be proven by contradiction: if there exists a city $\tilde{c}$ such that the distance from this city to the closest *best city* is greater than 2OPT, then (by construction of the greedy center selection algorithm) for every couple of *best cities* we have that $d(bc_i, bc_j) > 2OPT$; this means that we have a set of *k+1* cities (the k *best cities* and the city $\tilde{c}$) that are further than 2OPT from each other and then the problem cannot be solved using k antennae with optimal solution OPT (indeed, we need a larger OPT, or more antennae).

Since the distance between each *best city* bc_i and the closest antenna a_i is at most OPT and the distance between any city and its closest *best city* must be at most 2OPT, for the triangle inequality, $d(c_i, a_i) \leq d(c_i, bc_i) + d(bc_i, a_i) \leq 2OPT + OPT = 3OPT$.

3)

To prove that there is no feasible $\alpha - approximation$ with $\alpha < 3$, we show that we could use this algorithm to solve dominating-set in poly-time. Given the graph $G = (V, E)$ and k as an instance of dominating-set, we build an instance of center selection by associating an antenna a_i and a city c_i for every $v_i \in V$ (by doing this, we are duplicating the number of nodes). We connect every couple of cities with an edge of weight 2 and we do the same with the set of antennae. We then proceed by connecting every couple of nodes (c_i, a_j) with the following rules: $d(a_i, c_j) = 1$ if i=j or if $(v_i, v_j) \in E$, otherwise $d(a_i, c_j) = 3$. Since the distance between an antenna (a node a_i) and a city (a node c_j) is either 1 or 3, the optimum of the instance of the "extended center selection" problem (ECS-problem from now on) will be either 1 or 3.

We now can solve an instance of dominating set iff we can solve the relative instance of the ECS-problem with value 1: assuming that there is a solution for the ECS-problem of value 1, the set of antennae forms a dominating-set on the original instance (by construction, if the distance between two nodes in the ECS-problem instance is 1, then, in the original graph, either they are the same vertex or there is an edge between them). For the other side we can say that if we have a dominating set of size k, then we can take the related antennae and they will form a solution for the ECS-problem with value=1 (since they form a dominating set, there is an edge connecting every vertex of the dominating-set to any other vertex of the graph and this means that, by construction, in the ECS-problem's graph the antennae in the dominating set are connected to any city with an edge of weight 1).

If an $\alpha - approximation$ algorithm with $\alpha < 3$ existed, by using the aforementioned reduction, we would be able to find a solution better than 3OPT (since the suitable value for the instance of the ECS-problem are 1 and 3, finding a solution better than 3OPT means finding a solution of value 1) in poly-time. If this was possible, we could use this algorithm to find a solution of value 1 in poly-time, that is equivalent to find a dominating set of size k in poly-time, which is impossible unless $P = NP$.