Andrea Panceri 1884749, Francesco Sudoso 1808353

Exercise 1

Data: G=(V,E) undirected graph with weights on the vertices. For each v ∈ V the weight w_v=d(v) where d(v) is the degree of the vertex v.

Goal: Find a vertex cover set U with the minimal total weight possible.

1) Any greedy algorithm gives a 2-approximation to the best vertex cover.

We will prove that any greedy algorithm gives a 2-approximation to the optimal vertex cover by showing that the weight of the vertex cover found by the worst possible greedy algorithm is at most twice the weight of the optimal vertex cover.

Let U be the vertex cover found by the worst possible greedy algorithm, it must be a valid vertex cover that cover each edge of the graph, we take for each edge e=(u,v) ∈ E both the endpoints u and v. At the end it takes every vertex in the graph (excluding disjoint vertices) and returns it as the vertex cover. The set constructed is the worst possible because everytime take all vertices. We will show that the weight of U is at most twice the weight of the optimal vertex cover. We know that the weight of a vertex in U is equal to its degree, and the total weight of U is the sum of the weights of all the vertices in U. Thus, the weight of U is given by:

$$w(U) = \sum_{v \in V} w_v = \sum_{v \in V} d(v) = 2*|E| \quad (1)$$

Since every vertex in the graph is included in U, the above sum includes every edge in the graph (we also consider the disjoint vertices because they have zero weight, they don't impact the sum). This statement is true because the weight of each vertex is the degree of the vertex. This implies that we count the edges outgoing and incoming for each vertex, which is equal to |E|. Given that the graph is undirected we count two times each edge. For this reason, above sum is 2*|E|, because each edge e=(u,v) is considered one time by u and one time by v. On the other hand, the optimal vertex cover U* = OPT must cover every edge in the graph because it must be a valid vertex cover. The weight of U* is at least |E|, because the weight of each vertex in U* is equal to its degree, and the minimum weight of a vertex in U* is 1, which occurs when the vertex is incident to exactly one edge. Since every edge in the graph must be covered by at least one vertex in U*, the total weight of U* is at least |E|.

$$w(U^*) = OPT = \sum_{u \in U^*} w_u = \sum_{u \in U^*} d(u) \geq |E| \quad (2)$$

It is also true that:

$$2*OPT = 2 * \sum_{u \in U^*} w_u = 2 * \sum_{u \in U^*} d(u) \geq 2*|E| \quad (3)$$

Finally we arrive to this statement:

$$w(U) = \sum_{v \in V} w_v = \sum_{v \in V} d(v) = 2*|E| \leq 2 * \sum_{u \in U^*} w_u = 2 * \sum_{u \in U^*} d(u) = 2*w(U^*) = 2*OPT \quad (4)$$

This shows that the weight of U is at most twice the weight of U*, which means that any greedy algorithm gives a 2-approximation to the optimal vertex cover. This is guaranteed by the fact we use the degree of vertex like weight. The disequality in the last statement-(4) is guaranteed by the statement (3).

Andrea Panceri 1884749, Francesco Sudoso 1808353

Exercise 2

Data: n arbitrary cards, each with a different number, and n is an even number. You are allowed to flip the cards one by one and decide whether to select it or not. You can select only one card. When we select a card the game terminates.

1. We have elaborate this randomized strategy:

using the fact that n is even, we flip every time the first half of the card and save the maximal value of this card. We will use this value as a threshold for the remaining cards. This value obviously is completely random and changes in every game. The only thing that is deterministic is the fact that we flip always half of the cards:

- we flip the first half of the cards and save the greatest value of these cards in a variable *current max*.
- We flip the remaining cards one by one until we find a value that is greater than the current max. In the case this value exists, we select this value and terminate the game. If we arrive at the last card we take that card because it is the last.

This strategy is very simple but is very good in terms of probability of win. For taking the maximal value, we must consider that we don't have any information on the range of the cards and we don't make any assumptions on the cards that remain to be flipped.

2. Now we demonstrate that this algorithm achieves a probability of taking the max value of at least ¼:

We want to consider the probability of not winning, choosing a card with a value not equal to the maximal one. We have two cases in which we lose:

1) The first is when the maximum is in the first half of the cards that we flip.
2) The second is when the maximum is in the second half, but we take another value greater than the current max but less than the maximum value.

Pr[Max value V is in the first half] $= \frac{1}{N} * \frac{N}{2} = \frac{1}{2}$

This because the probability of value V to be the maximal value is $\frac{1}{N}$, and the number of possible cards that can be the maximal value in the first half is $\frac{N}{2}$.

Pr[We choose a value W < V the max value | Max value V is in the second half] = $\frac{1}{2} * \frac{\frac{n}{2}-1}{n} = \frac{n-2}{4n}$

This because the probability of the maximal value to be in the second half is like the one calculated before, and the probability of taking a value W < V is $\frac{\frac{n}{2}-1}{n}$ because we have $\frac{n}{2} - 1$ values that have this possibility given that only one is the max value and we divide by n because we must consider all the cards.

Like we said before the probability of not win is the sum of the probability of the two cases:

Pr[We don't choose the max value V] = $\frac{1}{2} + \frac{n-2}{4n} = \frac{3n-2}{4n}$

$\lim_{n \to +\infty}$ Pr[We don't choose the max value V] = $\lim_{n \to +\infty} \frac{3n-2}{4n} = \frac{3}{4}$

The probability of win:

Pr[we choose the max value V] = 1 - Pr[We don't choose the max value V] ≥ $\frac{1}{4}$

Using this algorithm, we take the maximum value with the probability of at least ¼.

Andrea Panceri 1884749, Francesco Sudoso 1808353

Exercise 3

Data: Directed graph G=(V,E), each vertex $v \in V$ has a weight w_v, an edge e=(u,v) is d-covered by a multi-set of vertices S if either u is in S at least once, or v is in S at least twice.

1. We do some consideration before formulating the IP: the vertex v can be in S at most twice, because can be inserted one time in S to cover an outgoing edge or two times to cover an incoming edge, no other possibility. We define two binary variables for each vertex $v \in V$: x(v) = 1 if vertex v is in S exactly one time, x(v) = 0 otherwise. y(v) = 1 if vertex v is in S exactly two times, x(v) = 0 otherwise. We ensure that x(v) and y(v) are never both one, and we consider the weight of each vertex in S the times that it is included in the set.

Objective function: $\min\left(\sum_{v \in V} (x(v) + 2*y(v))*w_v\right)$

s.t. $x(u) + y(u) + y(v) \geq 1$ $\quad \forall\ e=(u,v) \in E$

$x(v)*y(v) = 0$ $\quad \forall\ v \in V$

$x(v) \in \{0, 1\}$ $\quad \forall\ v \in V$

$y(v) \in \{0, 1\}$ $\quad \forall\ v \in V$

Like this we minimize the weight of the multi-set of vertices S that d-cover each edge with the first constraint.

2. Now we write the LP that relaxes the IP, because the IP is an NP-Hard problem and we don't know how to solve it in polynomial time, while the relaxed LP can be solved in polynomial time. All the multiset are still feasible solutions to the LP relaxation. The optimum to the LP relaxation is a lower bound to the optimum multiset set. We simply relax the constraint on the two binary variables x(v) and y(v):

Objective Function: $\min\left(\sum_{v \in V} (x(v) + 2*y(v))*w_v\right)$

s.t. $x(u) + y(u) + y(v) \geq 1$ $\quad \forall\ e=(u,v) \in E$

$x(v)*y(v) = 0$ $\quad \forall\ v \in V$

$0 \leq x(v) \leq 1$ $\quad \forall\ v \in V$

$0 \leq y(v) \leq 1$ $\quad \forall\ v \in V$

3. The last step is to define a rounding scheme that guarantees a 2-approximation to the best multi-set. We want to find an optimal solution x*, y* to the relaxed problem in polynomial time. Round x*,y* to a $\bar{x}, \bar{y}$:

- $\bar{x}(v) = 1$ if $x^*(v) \geq ½$ or $\bar{x}(v) = 0$ if $x^*(v) < ½$
- $\bar{y}(v) = 1$ if $y^*(v) \geq ½$ or $\bar{y}(v) = 0$ if $y^*(v) < ½$

The solution $\bar{x}$ and $\bar{y}$ are feasible:

$$\forall\ e=(u,v) \in E,\ \bar{x}(u) + \bar{y}(u) + \bar{y}(v) \geq 1 \text{ and } \forall\ v \in V\ \ \bar{x}(v)*\bar{y}(v)=0$$

since either:

- $x^*(u) \geq ½$ or $y^*(u) \geq ½$ or $y^*(v) \geq ½$
- when $x^*(v) \geq 0$ then $y^*(v) = 0$ or the viceversa $\forall\ v \in V$, this is fundamental.

The solution is 2-approximation because:

$$\sum_{v \in V} (\bar{x}(v) + 2\ *\ \bar{y}(v))\ *\ w_v \leq \sum_{v \in V} (2x^*(v) + 4y^*(v))\ *\ w_v \leq 2\ *\ \sum_{v \in V} (x^*(v) + 2y^*(v))\ *\ w_v \leq 2\ *\ OPT$$

since $\bar{x}(v) \leq 2x^*(v)$ and $\bar{y}(v) \leq 2y^*(v)$

Andrea Panceri 1884749, Francesco Sudoso 1808353

Exercise 4

Data: NE is the set of all the Nash equilibrium states, and s_{OPT} is the state with the optimal Social Cost (SC). We define the Price of Stability like PoS = $min_{s \in NE} \frac{SC(s)}{SC(s_{OPT})}$. We also define the Rosenthal's potential function like this $\Phi(\alpha) = \sum_{r \in R} \sum_{i=1}^{\#(r,\alpha)} c_r(i)$. We consider a congestion game in which the cost functions c_r are such that no resource is ever used by more than λ players.

Question: Use Rosenthal's potential to show that the Price of Stability of such a game is at most λ.

Answer: We can start with the hint of the problem which explains that in case a state is not a Nash equilibrium, there is at least one player that by deviating could decrease the potential function. The potential function in a state that is a Nash equilibrium is the global minimum of the function, because no player can deviate, thus:

$$\Phi(\alpha) \leq \Phi(\alpha^*) \quad (1)$$

where α^* is the optimal state that could not be equal to the Nash equilibrium state, and α is a state that belongs to NE . We define $k = \#(r, \alpha)$. Another important consideration is this:

$$SC(\alpha) = \sum_{r \in R} c_r(k) * k \geq \Phi(\alpha) = \sum_{r \in R} \sum_{i=1}^{k} c_r(i) \Rightarrow c_r(k) * k \geq \sum_{i=1}^{k} c_r(i) \Rightarrow SC(\alpha) \geq \Phi(\alpha) \quad (2)$$

We can conclude that the social cost in a state α is always greater or equal to the potential function of that state, due to the property of the cost function c_r to be monotonically increasing.

In addition, we use the fact that $k = \#(r, \alpha) \leq \lambda$ for proving the following statement:

$$SC(\alpha) = \sum_{r \in R} c_r(k) * k \leq \lambda * \Phi(\alpha) = \lambda * \sum_{r \in R} \sum_{i=1}^{k} c_r(i) \Rightarrow c_r(k) * k \leq \lambda * \sum_{i=1}^{k} c_r(i) \Rightarrow SC(\alpha) \leq \lambda * \Phi(\alpha) \quad (3)$$

Using the following simple mathematical properties:

1) $c_r(\lambda) * \lambda \leq \lambda * \sum_{i=1}^{\lambda} c_r(i) = \lambda * c_r(\lambda) + \sum_{i=1}^{\lambda-1} c_r(i)$

   $\sum_{i=1}^{\lambda-1} c_r(i) > 0$ because the cost function is always positive.

   This statement is also true for $k = 1, \dots, \lambda$.
2) The max value of k is λ.
3) The monotonically increasing property of c_r.

We have demonstrated the previous statement (3).

Now we can recall all previouses statements for arrive to the solution of the problem:

$$SC(\alpha) \leq \lambda * \Phi(\alpha) \leq \lambda * \Phi(\alpha^*) \leq \lambda * SC(\alpha^*) \Rightarrow SC(\alpha) \leq \lambda * SC(\alpha^*) \Rightarrow PoS = min_{\alpha \in NE} \frac{SC(\alpha)}{SC(\alpha^*)} \leq \lambda$$

We have demonstrated the statement of the exercise as follows:

1) $SC(\alpha) \leq \lambda * \Phi(\alpha)$: we use the (3)-statement.
2) $\lambda * \Phi(\alpha) \leq \lambda * \Phi(\alpha^*)$: we use the (1)-statement.
3) $\lambda * \Phi(\alpha^*) \leq \lambda * SC(\alpha^*)$: we use the (2)-statement.
4) $SC(\alpha) \leq \lambda * SC(\alpha^*)$: we have considered only the first and last terms of disequations.
5) $\frac{SC(\alpha)}{SC(\alpha^*)} \leq \lambda$ and this is valid also for the state α s.t. $min_{\alpha \in NE} SC(\alpha)$.
6) $PoS \leq \lambda$.

Andrea Panceri 1884749, Francesco Sudoso 1808353

Exercise 5

Data: Undirected graph G=(V= {v_1, …, v_n}, E), n! possible orderings of these vertices. Pick one such ordering uniformly at random $\sigma = (\sigma_1, \ldots, \sigma_n)$. We have this algorithm:

Begin with S = ∅. Then, at each step (for i = 1 to n), if for all $u \in S$, $(u, v\sigma_1) \notin E$, add $v\sigma_i$ to S.

d is the maximum degree of a vertex of V .

Question: Prove that the proposed algorithm achieves an independent set with expected value of at least 1/d fraction of the optimal solution.

Answer: We denote the optimal solution for the independent set with S* and with |S*|= OPT the size of the maximum independent set. S is the solution of the proposed algorithm and |S| is its size.

We consider the elements of S = {s_1 , ... , s_k} and |S| = k. For each s_i, the neighbors of such a vertex are at most d, because d is the maximum degree.

We add s_i and his neighbors to a set N_i = {s_i ∪ Neighbors of s_i }. The size of this set is at most $|N_i| = d + 1$, because it contains s_i and at most other d vertices connected to s_i .

Now we can argue some strong condition: $|S^* \cap N_i| \leq d$. (1)

This is true because the optimal set contains the vertex s_i or contains some neighbors of s_i , at most this number of vertices is d, not possible otherwise for the definition of independent set. The intersection between the two sets for this reason is at most d.

So we can iterate this for the k vertices of S, and write this statement:

$$OPT = \sum_{i=1}^{k} |S^* \cap N_i| \leq \sum_{i=1}^{k} d = k * d = |S| * d \quad (2)$$

$$OPT \leq |S| * d \quad (3)$$

$$|S| \geq \frac{OPT}{d} \quad (4)$$

Like this we prove that the set S that we constructed with the algorithm will achieve a $\frac{1}{d} - approximation$ of the optimal solution S*.

$$\sum_{i=1}^{k} |S^* \cap N_i| \leq \sum_{i=1}^{k} d$$

This disequality is a consequence of statement (1), because it is the same with summation on both sides.

$$OPT = \sum_{i=1}^{k} |S^* \cap N_i|$$

This is valid because in the optimal solution we will have some vertices of S or some of the neighbors of vertices of S. So we can sum all $|S^* \cap N_i|$ for i =1, …, k and obtain the optimal independent set, it can't contain other vertices.

With the algorithm proposed we obtain always an independent set with at least $\frac{1}{d}$ of the vertices of the optimal independent set.