Ex. 1

Yuval Gitlitz & Oren Roth

28.4

1. a) Let $G = (V, E), w$ be our graph and the weight function on the edges respectively. We will create bipartite graph $G' = (V \times \{0\}, V \times \{1\}, E')$ where,
$$E' = \{((u, 0), (v, 1)) : (u, v) \in E\}$$
With weight function $w' : E' \to R$, s.t. $w'(((u, 0), (v, 1))) = w((u, v))$. We will run weighted prefect matching and receive M. We will build the cycle cover accordingly to M, cycle by cycle. c_0 will be constructed by taking and delete an edge $((u, 0), (v, 1))$ in M and add (u, v) to c_0, go on by take $((v, 0), (w, 1))$ in M, remove it from M and add (v, w) to the cycle until we will reach a node which is matched to $(u, 1)$. By then we will finish one cycle and if there are more edges in M we will construct a new cycle c_1 and so on until there are no more edges to delete in M.
b) The algorithm:

   - Find min cost cycle cover - denoted by $C = (c_1, \ldots, c_k)$. For every $i \in [k]$, define $e_i = (u_i, v_i)$ as an edge in c_i.
   - $G \leftarrow \{(u_k, v_1)\}$
   - for $i = 1$ to $k - 1$ do:
     - $G \leftarrow G \cup (c_i \setminus \{e_i\} \cup \{u_i, v_{i+1}\})$
   - $G \leftarrow G \cup (c_i \setminus \{e_i\} \cup \{u_i, v_{i+1}\})$

   Proof. We will show :

   I G is Hamiltonian cycle.

   II cost G is at most $\frac{4}{3}OPT$.

   I We will show the edges in G admit Hamiltonian cycle. We start by v_1 and go throug edges of cycle c_1 until the node u_1 than take the edge u_1, v_2 and continue in this fashion until reaching node u_k, then taking the edge $\{(u_k, v_1)\}$ and we done,

   II $cost(C) \leq OPT$ because the optimal solution is feasible solution for the cycle cover problem. As each cycle is at least of size of 3 we have

that $k \leq \frac{|V|}{3}$. G replace k edges of size at least 1 with k edges of size at most 2, then:

$$G \leq cost(C) + k \leq cost(C) + \frac{|V|}{3} \leq OPT + \frac{|V|}{3} \leq \frac{4}{3}OPT$$

And the last inequality is due to the fact that the optimal solution visits $|V|$ edges of weight one at least.

□

2. (a) We build MST $T = (R, E')$ on the sub graph which includes only nodes in R. Our algorithm will return T which is also a feasible solution. We will show $c(T)$ is at most $2OPT$. Let $\tilde{T} = (\tilde{V}, \tilde{E})$ be the steiner tree which has $c(\tilde{T}) = OPT$. $c(\tilde{T}) = \sum_{v \in \tilde{V}} c(v) + \sum_{e \in \tilde{E}} c(e)$. In the same way as we showed in class we can have that:

$$2 \cdot \sum_{e \in \tilde{E}} c(e) \geq \sum_{e \in E'} c(e)$$

and since $\sum_{v \in R} = 0$ we conclude:

$$c(T) = \sum_{v \in R} c(v) + \sum_{e \in E'} c(e) = \sum_{e \in \tilde{E}} c(e) \leq 2OPT$$

(b) Assume towards contradiction that there is exists a $(c \cdot ln|R|)$-approximation algorithm, we will show how to build a reduction based $O(logn)$-approximation algorithm for set cover and we will arrive to contradiction.

The reduction algorithm:

i. Given $X = (U, S = \{S_1, \ldots S_m\})$ input for set cover, build the following steiner tree input, $X' = (G = (V, E), R, w)$ where:

$$\begin{aligned}
&V = U \cup S \\
&E = (S \times S) \cup \{(S_i, e_j) : e_j \in S_i, S_i \in S\} \\
&R = U \\
&\forall e \in E : \quad w(e) = 0 \\
&\forall S_i \in S : \quad w(S_i) = w_i \\
&\forall e_i \in U : \quad w(e_i) = 0
\end{aligned}$$

ii. Run the $(c \cdot ln|R|)$-approximation algorithm on G, R, w and receive T.

iii. Return $I = \{S_i : S_i \in V(T); S_i \in S\}$.

We will state two useful lemmas:

Lemma 1. *Given $\tilde{T}$ solution to X', $\tilde{I} = \{S_i : S_i \in V(\tilde{T}); S_i \in S\}$ is a feasible solution for X.*

Proof. We set $R = U$, and because each node in R is only connected to their sets, hence because is T connected the only way to saturate all the terminal is by taking sets which include all of them - and we conclude I is a valid solution to the set cover problem. □

Lemma 2. *Given $\tilde{T}$ solution to X', $\tilde{I} = \{S_i : S_i \in V(\tilde{T}); S_i \in S\}$ has the same weight of $\tilde{T}$ in X.*

Proof. The weight of nodes in S is the same as the weight of the set cover weights, all the other nodes and edges are of weight zero. Therefore:

$$w(\tilde{T}) = w(\{S_i : S_i \in V(\tilde{T}); S_i \in S\}) = w(\tilde{I})$$

□

Claim 3. *The algorithm is $O(logn)$-approximation algorithm for set cover.*

Proof. By Lemma 1 I is feasible solution and by Lemma 2 we know $w(I) = w(T)$. Denote by $O_{steiner}, O_{set-cover}$ the optimal solutions values of X', X respectively. We conclude:

$$w(I) = w(T) \overset{(i)}{\leq} (c \cdot ln|R|) \cdot O_{steiner} = (c \cdot ln|R|) \cdot O_{set-cover}$$

(i) is due to our assumption and the last equality is due to Lemma 2. □

As $|R| = n$ we found an $O(logn)$-approximation algorithm for set-cover which accordingly to what we learn in class could happen only if $P = NP$.

(c)

3.

4. (a) Let $G = (V, E)$ be a graph we will show the claim holds by induction on $|V|$. Base: $|V| = 0$ trivial. Assume that when $|V| < n$ the claim holds. Let be G be a graph with maximum degree Δ, with $|V| = n$ and let $v \in V$. Let $G' = G - v$. $|V(G')| = n - 1$ and its maximum degree at most Δ. We use the induction hypothesis in order to color G' with $\Delta + 1$ colors. Use the same coloring used for G' in G for all the vertices except v. For v, it has at most Δ neighbors and it can be colored using a different color than its neighbors. We used at most $\Delta + 1$ to color the vertices in G hence the claim holds.

The algorithm for finding $(\Delta + 1)$-coloring will work in a greedy fashion each time choose an uncolored node and color it with an available color. As the maximum degree is Δ we know we can do it with $\Delta + 1$ colors.

Next we will show that a bipartite graph is two colorable. Let $G = (A, B, E)$ be a bipartite graph. We color all the vertices in A using the first color and all the vertices in B using the second color. All the edges in A are disjoint hence we don't have two vertices which use the first color which are connected. The same applies for B and the second color.

(b) **Algorithm for finding** $O(\sqrt{(n)}) - coloring$**:**

i. While there exist $v \in V(G)$ such that $\deg(v) \geq \sqrt{n}$
   A. Color its neighbors using two colors
   B. $G \leftarrow G - N(v)$
ii. color G using $\sqrt{(n)} + 1$ colors

Claim 4. *The algorithm run in polynomial time*

Proof. First, let us show that step A can be done in polynomial time. The neighborhood of any vertex v in the graph can be two colored because each subgraph is three colored, and if we used one color for v, the neighborhood can be two colored. Two colored subgraph is also a two bipartite graph, hence we can use the previous question to color it using 2 colors.

The loop in step i runs at most $\sqrt{n}$ times because each iteration, we remove at least $\sqrt{n}$ vertices from G. Additionally, each iteration run polynomial time. Hence, the total run time of step i is polynomial.

Step ii runs in polynomial time using the algorithm from the previous section. □

Claim 5. *The algorithm is using* $O(\sqrt{n})$ *colors*

Proof. In each iteration of loop i, we use two colors. There are at most $\sqrt{n}$ iteration, hence for step i we use $2\sqrt{n} = O(\sqrt{n})$ colors. For step i we used $\sqrt{n} + 1$ colors. For that reason, the total number of colors used by the algorithm is $O(\sqrt{n})$. □

(c) **Algorithm for finding** $O(n^{\frac{2}{3}}) - coloring$**:**

i. While there exist $v \in V(G)$ such that $\deg(v) \geq n^{\frac{2}{3}}$
   A. Color $n^{\frac{2}{3}}$ of his neighbors using $O(n^{\frac{1}{3}})$ colors.
   B. $G \leftarrow G - N(v)$
ii. color G using $n^{\frac{2}{3}} + 1$ colors

Claim 6. *The algorithm run in polynomial time*

Proof. First, let us show that step A can be done in polynomial time. The neighborhood of any vertex v in the graph can be three colored because each subgraph is 4-colored, and if we used one color for v, the neighborhood can be three colored. Three colored subgraph of size $n^{\frac{2}{3}}$ can be colored with using the previous algorithm with $n^{\frac{1}{3}}$.
The loop in step i runs at most $n^{\frac{1}{3}}$ times because each iteration, we remove at least $n^{\frac{2}{3}}$ vertices from G. Additionally, each iteration run polynomial time. Hence, the total run time of step i is polynomial.
Step ii runs in polynomial time using the algorithm from section (a). □

Claim 7. *The algorithm is using $O(n^{\frac{2}{3}})$ colors*

Proof. In each iteration of loop i, we use $n^{\frac{1}{3}}$ colors. There are at most $n^{\frac{1}{3}}$ iteration, hence for step i we use $n^{\frac{2}{3}}$ colors. For step i we used $n^{\frac{2}{3}} + 1$ colors. For that reason, the total number of colors used by the algorithm is $O(n^{\frac{2}{3}})$. □