Assignment1

Praneeth Kacham

1 Question-2

2 Question-3

Let V be the set of vertices and S_1 and S_2 be the parition of V. Pick an arbitrary vertex and put it into S_1. Now iterate over the remaining vertices and one after another put them into S_1 or S_2 depending on which of the configurations gives greater cut size based on the vertices added to S_1 and S_2 till that point. This is a 1/2-approximation. Notation: v_i be the vertex picked in ith iteration, $S_1(i), S_2(i)$ be the sets S_1 and S_2 at the end of the ith iteration, $V(i) = S_1(i) \cup S_2(i)$ and $G(i)$ be the graph induced by the vertices $V(i)$.
Claim: Partition $(S_1(i), S_2(i))$ is a 1/2-approximation for $G(i)$.

Proof. Proof by induction on i.

True for $i = 1$ as the optimal partition has weight 0.

Assume that the claim is true for all $k \leq i$.

Without loss of generality, assume that $v_i \in S_1$. Let $(A, V(i) - A)$ be the optimal partition for $G(i)$ and $v(i) \in A$. We have $OPT(G(i)) = wt((A, V(i) - A)) = wt((A - v(i), V(i) - A))$ + contribution of v_i. But $wt((A - v(i), V(i) - A)) \leq OPT(G(i-1))$ and contribution of $v_i \leq$ weight of edges incident on v_i in $G(i)$. So,

$$OPT(G(i)) \leq OPT(G(i-1)) + \text{weight of edges incident on } v_i \tag{1}$$

By the induction hypothesis we have $wt(S_1(i)-v_i, S_2(i)) = wt(S_1(i-1), S_2(i-1)) \geq OPT(G(i-1))/2 \geq wt((A-v(i), V(i)-A))/2$. We also have $wt(v_i, S_2(i)) \geq$ (weight of edges incident on v_i)/2 as v_i is added to S_1 by the greedy algorithm. So, $wt(S_1(i), S_2(i)) = wt(S_1(i) - v_i, S_2(i)) +$ contr. of $v_i \geq OPT(G(i-1))/2 +$ (wt. of edges incident on v_i)/2 $\geq OPT(G(i))/2$. □

Thus the partition $(S_1(n), S_2(n))$ is a 1/2-approximation for the graph $G(n) = G$.

The generalized algorithm for k partitions is as follows. Start with $S_1, S_2, \ldots, S_k = \phi$. Iterate over the vertices v_i. Put the vertex v_i into the set S_j such that

$$j = \text{argmax}_j wt(S_1, \ldots, S_{j-1}, S_j \cup v_i, S_{j+1}, \ldots, S_k) \tag{2}$$

So,

$$\text{contibution of } v_i \text{ at the end of ith iteration} = \max_j wt(S_1, S_2, \ldots, S_{j-1}, v_i, S_{j+1}, \ldots, S_k) \tag{3}$$

$$\geq \frac{1}{k} \sum_j wt(S_1, S_2, \ldots, S_{j-1}, v_i, S_{j+1}, \ldots, S_k) \tag{4}$$

$$\geq \frac{1}{k}(k-1)\text{wt. of edges incident on } v_i \tag{5}$$

So, given that $wt(S_1(i-1), S_2(i-1) \ldots, S_k(i-1))$ is $\geq (1-1/k)OPT(G(i-1))$ we have $wt(S_1(i), \ldots, S_j(i), \ldots, S_k(i)) = wt(S_1(i-1), S_2(i-1), \ldots, S_j(i-1) \cup v_i, \ldots, S_k(i-1)) = wt(S_1, S_2, \ldots, S_j, \ldots, S_k) +$ contr. of $v_i \geq (1-1/k)OPT(G(i-1)) + (1-1/k)$wt. of edges incident on $v_i \geq (1-1/k)OPT(G(i))$. Hence $(S_1(n), S_2(n), \ldots, S_k(n))$ is $(1-1/k)$ approximate partition of $G(n) = G$.

3 Question-4

Let $S = \{1, 2, 3, \ldots, k\}$. And each of them have a coverage requirement of $\alpha_i \in \mathbb{Z}^+$. Let $\sum_i \alpha_i = n$. The greedy algorithm is as follows:

1. $\beta_i = \alpha_i$ forall $i \in S$ and $r_i = 0$ $\forall S_i$.
2. if $\exists i$ $\beta_i > 0$, continue else exit
3. find j such that $\frac{w(S_j)}{|S_j \cap \{i|\beta_i > 0\}|}$ is minimum.
4. $r_j := r_j + 1$ and $\beta_i := \beta_i - 1$ $\forall i \in S_j$
5. Go to step 2

The above algorithm is a $\ln n$-approximate algorithm. Let $n_i = \sum_j \beta_j$ before ith iteration. So, $n_1 = n$ and $n_i - n_{i+1}$ is the number of elements covered in ith iteration. Let S_k be the set chosen in ith iteration. Optimum solution can cover n_i elements with cost OPT. So, there should be a set in OPT which can cover some elements with average cost less than OPT/n_i. But average cost of S_k should be even less. Hence,

$$w(S_k) \leq OPT * (n_i - n_{i+1})/n_i \tag{6}$$

Summing the above inequality over all the iterations, we have

$$\sum_j r_j wt(S_j) \leq OPT * H_n \tag{7}$$

So, this algorithm is a $\ln n$-approximation.

4 Question-5

Consider three sets of vertices $\{r\}$, $\{v_1, v_2, \ldots, v_m\}$ one for each of the sets S_i and $\{u_1, u_2, \ldots, u_n\}$ one for each of the elements of set S. Construct a graph G with vertices as the union of $\{r\}$, $\{v_1, v_2, \ldots, v_m\}$ and $\{u_1, u_2, \ldots, u_n\}$ with the edges as follows, $r \to v_i$ for all $i = 1..n$ with cost of edge $r \to v_i = wt(S_i)$ and the edges $v_i \to u_j$ $\forall$ i, j $such$ $that$ $j \in S_i$ of cost 0. Set the vertices $r, \{u_1, u_2, \ldots, u_n\}$ as required and r as the special vertex and the vertices $\{v_1, v_2, \ldots, v_m\}$ as steiner vertices. Any tree with root r and containing the vertex u_j should have an edge to v_k such that there is an edge.
Let $T = (V', E')$ is a steiner tree. Using this we shall construct a set-cover. Consider the set-cover $\mathcal{S} = \{S_j | r \to v_j \in E'\}$.
Claim: $\mathcal{S}$ is a vertex cover and $cost(\mathcal{S}) = wt(T)$.

Proof. Consider an element $e_i \in S$. There is a corresponding vertex u_i in the vertex set of G and u_i is a required vertex. So, there exists a vertex v_k such that $r \to v_k$ and $v_k \to u_i$ which implies $S_k \in \mathcal{S}$ and $e_i \in S_k$. So, $e_i \in \cup_{S_j \in \mathcal{S}} S_j$. As e_i is an arbitrary element of $\mathcal{S}$, we have $\mathcal{S}$ is a set cover. $cost(\mathcal{S}) = \sum_{S_j \in \mathcal{S}} wt(S_j) = \sum_{v_j : r \to v_j \in E'} wt(r \to E') = wt(T)$. So, cost of the set-cover $\mathcal{S}$ is $wt(T)$. $\square$

Let $\mathcal{S}$ be a vertex cover. We shall construct a steiner tree T. Consider the graph $T = (V, E)$ with V as union of $\{r\}$,$\{v_j | S_j \in \mathcal{S}\}$ and $\{u_1, u_2, \ldots, u_n\}$ and edges as union of $\{r \to v_j | v_j \in V\}$ and $\{v_j \to u_k | e_k \in S_j\}$. Remove edges if there are morethan one edges incoming to a vertex of form u_k. Given that $\mathcal{S}$ is a setcover, we can see that the tree T is a steiner tree and cost of tree T is equal to weight of set cover $\mathcal{S}$.

From above,

$$OPT(SteinerTree) = OPT(SetCover) \tag{8}$$

Hence, given a $O(\log(n))$ approximation algorithm for steiner-tree, we can solve the steiner tree problem corresponding to the setcover problem and give $O(log(n))$ approximation to set-cover problem.