Do not look up research papers on the Web. Work in a group of 2. Submit one homework solution per group.

Total Point: 150

1. (i) Consider the adjacency list representation for $G = (V, E)$. Keep a count on total number of edges and vertices. Keep an array degree such that $\text{degree}(i)$ gives the current degree of vertex $i$. Maintain another array buckets such that $\text{bucket}(i)$ contains a list of vertices with current degree at most $i$. Initialize $\text{bucket}$ so that $\text{bucket}(i)$ contains all vertices with degree exactly $i$ to start with. Maintain a pointer to the minimum index $\text{min}$ such that $\text{bucket}(\text{min})$ is non-empty and $\text{bucket}(j)$ is empty for all $0 < j < \text{min}$.

Remove a vertex $i$ from $\text{bucket}(\text{min})$, check if $\text{degree}(i) = \text{min}$. If not delete $i$. If $\text{bucket}(\text{min})$ is empty, update $\text{min} = \text{min} + 1$. Repeat this process until a vertex $i$ is found from $\text{bucket}(\text{min})$ such that $\text{degree}(i) = \text{min}$.

Compute density of the remaining graph by subtracting $\text{degree}(i)$ from the total edge count and 1 from total vertex count. For every neighbor $j$ of $i$, decrement $\text{degree}(j)$ by 1, and insert $j$ in $\text{bucket}(\text{degree}(j))$. If $\text{degree}(j) < \text{min}$, update $\text{min} = \text{degree}(j)$.

Repeat the process until all vertices are deleted.

$\text{degree}$ array always maintains the current degree of every vertex. Each vertex $i$ is inserted in the bucket and deleted at most the number of its original neighbors. Hence, the total time required is $O(|V| + |E|)$.

(ii) Let $OPT$ denote the optimum densest $k$ subgraph. Since we pick $k/2$ highest degree vertices in $U$, total number of edges incident on $U$, denoted $E'(U)$ is at least as high as total number of edges incident on top-$k/2$ vertices in $OPT$. Hence $|E'(U)| \geq \frac{|E(\text{OPT})|}{2}$.

Let $X$ and $Y$ denote the number of edges with both end points in $U$ and just one end point in $U$ respectively. Hence $E'(U) = X + Y$. Now we pick $k/2$ vertices from $V/U$ with maximum number of edges incident on $U$. Therefore, the total number of edges induced by our chosen subgraph is at least $X + \frac{k}{2n}Y \geq \frac{k}{2n}|E'(U)| \geq \frac{k}{4n}|E(\text{OPT})|$.

2. (i) We show the greedy algorithm for the unweighted set cover achieves an $\ln n$ approximation, where $OPT$ is the minimum number of sets selected by an optimum algorithm. If all sets have size at most $d$, then $OPT \geq n/d$, so $d \geq n/OPT$ and we get a $(\ln d + 1)$-approximation.

Let $k = OPT$ and let $E_i$ be the set of elements not yet covered after step $i$, with $E_0 = U$. The $k$ sets of the optimal solution cover every elements of $E_t$. Since the greedy algorithm always picks the set with largest number of uncovered elements, the set picked in the $t + 1$th iteration covers at least $\frac{|E_t|}{k}$ elements. Hence $|E_{t+1}| = (1 - \frac{1}{k}) |E_t|$. Therefore, by induction,
\[ |E_t| \leq n \left( 1 - \frac{1}{k} \right)^t. \] Consider when \(|E_t| < k\). We have
\[
\left( 1 - \frac{1}{k} \right)^t < \frac{k}{n}
\]
\[ e^{-\frac{t}{k}} \leq \frac{k}{n} \]
\[ t \leq k \ln \frac{n}{k} \]

Thus, after \(k \ln \frac{n}{k}\) steps there remain only \(k\) elements. Each subsequent step removes at least one element. So the greedy algorithm selects at most \(k(1 + \ln \frac{n}{k})\) sets.

(ii) Consider a graph with \(k\) layers \(L_1, L_2, ..., L_k\), such that \(L_i\) contains \(\lceil \frac{n}{i} \rceil\) vertices. Connect the \(j\)th vertex in \(L_i\) to \(i\) vertices indexed \((j-1)i+1\) to \(ji\) in \(L_1\). An optimal vertex cover picks the \(k\) vertices from \(L_1\). While the greedy algorithm may pick the single vertex from \(L_k\), then the vertices in \(L_{k-1}\) and so on upto vertices in \(L_2\). The total number of vertices picked is \(O(k \log k)\). Since the total number of vertices is \(O(k \log k)\), we get an approximation factor of \(O(\log n)\).

(iii) Easy.

3. Given a bipartite graph \(G = (A \cup B, E)\) direct all edges from \(A\) to \(B\) and assign a weight of \(\infty\). Add a source node \(s\), and connect it to all nodes in \(A\) with a directed edge with capacity of one. Add a sink node \(t\), and connect all nodes in \(B\) to the sink node with a directed edge of capacity \(1\). Now find the maxflow from \(s\) to \(t\) to obtain a \(s-t\) min-cut \(\{s, A_1, B_1\}, \{t, A_2, B_2\}\), \(A_1, A_2 \subseteq A\), and \(B_1, B_2 \subseteq B\).

There exists a \(s-t\) cut with value \(|A|\). Hence, the cut cannot cross any weight \(\infty\) edge. Clearly then the value of min-cut/max-flow is \(|A_2| + |B_1|\). This is same as the value of maximum matching in \(G\) (easy to check). We claim \(A_2 \cup B_1\) is also a vertex cover. This is true, because if there is an edge from \(A_1\) to \(B_2\), then the value of the cut would be \(\infty\).

4. To prove \((a) \rightarrow (b)\), apply (a) to obtain \(f(A + v) + f(B) \geq f(B + v) + f(A)\).

To prove \((b) \rightarrow (a)\), apply (b) repeatedly by adding elements from \(A \setminus B\) to \(A \cap B\) and \(B\) respectively. We get
\[
f((A \cap B) + (A \setminus B)) - f(A \cap B) \geq f(B + (A \setminus B)) - f(B)
\]
\[
\text{or, } f(A) - f(A \cap B) \geq f(A \cup B) - f(B)
\]

(ii)(a) Let \(B \setminus A = \{v_1, v_2, ..., v_t\}\).
\[
f(B) - f(A) = [f(A + v_1) - f(A)] + [f(A + v_1 + v_2) - f(A + v_1)] + ... + [f(B) - f(B - v_t)]
\]
\[
\leq \sum_{i=1}^{t} f(A + v_i) - f(A) \leq |B \setminus A|[f(A + v_{max}) - f(A)]
\]

where \(v_{max}\) maximizes \(f(A + v_{max})\).

(ii)(b) is similar.