Name: Kechen Zhao 95398

Problem 1

Define $d_{max} = \max_i\{d_i\}$, where d_i = # edges of vertex i in the whole graph.

Then we will first prove that for any subgraph G_s of G, we have

$$\max\{\text{density}(G_s)\} \le \frac{d_{max}}{2}$$

proof: $\text{density}(G_s) = \frac{|E_s|}{|S|} = \frac{\#\text{edges}}{\#\text{vertex}}$

Since d_{max} = maximum number of edges of a vertex in the whole graph G, then

$\#\text{edges} \le d_{max} \cdot \#\text{vertex}$ for any G_s

However, each edge must be formed between 2 vertices, then when counting total number of edges, we count edge twice by $d_{max} \cdot \#\text{vertex}$, so

$\Rightarrow |E_s| \le \frac{1}{2} d_{max} \cdot |S| \Rightarrow \frac{|E_s|}{|S|} \le \frac{d_{max}}{2}$ $\forall G_s \in G$

Now we have an upper bound for the density of each subgraph of G. We then define d_{max_a} be the maximum density value for all subgraphs generated by the Greedy Peeling Algorithm.

Let v_{min} be the first vertex that been removed. After remove v_{min}, density will become

$$d(G_s) = \frac{\#\text{edges in } G - \text{degree of } v_{min}}{\#\text{nodes in } G - 1} \le d_{max\_a}$$

From above, we can get that degree of $v_{min} \le 2 \cdot \frac{|E_s|}{|S|} = d_{max\_a}$

$$d(G_s) = \frac{\#\text{edges in } G - \text{degree of } v_{min}}{\#\text{nodes in } G - 1}$$

$$(\text{worst case}) = \frac{\#\text{edges in } G - d_{max\_a}}{\#\text{nodes in } G - 1} > \frac{\#\text{edges in } G - d_{max\_a}}{\#\text{nodes in } G}$$

$$= \frac{\#\text{edge in } G}{\#\text{nodes in } G} - \frac{d_{max\_a}}{\#\text{nodes in } G} > \frac{\#\text{edge in } G}{\#\text{nodes in } G} - d_{max\_a} \le d_{max\_a}$$

$$\Rightarrow \max\left\{\frac{\#\text{edge in } G}{\#\text{nodes in } G}\right\} = \frac{d_{max}}{2} \Rightarrow \frac{d_{max}}{2} - d_{max\_a} \le d_{max\_a}$$

$$\Rightarrow \frac{d_{max}}{2} \le 2\, d_{max\_a}$$

$\Rightarrow$ Proved.

Problem 2 : $O(|S| \cdot \log|S|)$

Problem 3 Name: Kechen Zhao 9/13/98

To achieve this bound. we need to make sure that delete-min should only take O(1) time.

Given a graph with n vertices and m edges, store the vertices that have the same degrees in to a list, then arrange the lists as values of a dictionary, whose key values are the unique degrees in the graph:

Dictionary = {"1": [v_1, v_2, v_3, v_4, ...], "2": [v_5, v_6, v_7, v_8, ...], "3": [v_9, v_{10}, v_{11}, v_{12}, ...], ...}

By using this data structure:

- searching the vertex with the smallest degree and removing it will only take O(1) since we can just take an arbitrary vertex from the first non-empty list;
- to update the degrees of the neighbors of the removed vertex, we can just move them from their current list to the previous one which with the key value one less than the current degree; there are total # degree of removed vertex nodes need to be updated;
- there are total |V| loops, and we need to update the position of vertices in dictionary |E| times in total |V| loops, and the rest of the operations all take O(1), so the overall running time is bounded by O(|V|+|E|) = O(n+m).

Problem 4

- Except the source node, set S will contain the edge-nodes that formed by the edges and the vertex-nodes that formed by the vertices which remain in the densest subgraph;
- except the sink node, set t will contain the edge-nodes that formed by the edges and the vertex-nodes that formed by the vertices which need to be deleted from the original graph in order to achieve the densest subgraph.
- 2 types of edges will across the cut:
  1. edges between source node the vertex-nodes that formed by the vertices which need to be deleted from the original graph in order to achieve the densest subgraph;
  2. edges between sink node and the edge-nodes that formed by the edges which remain in the densest subgraph;

Name : Kechen Zhao 857398

Problem 5

The minimum s-t cut has a capacity less than $\lambda \cdot |E| = \lambda \cdot m$ means that at least one edge in the original graph stays in the densest subgraph, the densest subgraph exists, so the density of the densest subgraph is at least the density of the original graph.

Problem 6

Since finding minimum cut of network flow can help to find the densest graph, we can construct an algorithm to do the following steps:

1. Construct a flow network based on the original graph; $O(m+n)$
2. Use Ford-Fulkerson to find the maximum flow and min-cut; $O(Cm)$
3. Vertices and edges that their corresponding vertex-nodes and edge-nodes stay in the set $S \setminus \{s\}$ will form the densest subgraph.

Since Ford-Fulkerson runs in polynomial time, constructing a flow network takes linear time, so the total complexity will be in polynomial time.